

A STIGMERGY-DRIVEN MULTI-AGENT FRAMEWORK FOR INTELLIGENT TASK ORCHESTRATION

EVA VELI

Thesis supervisor
JORDI PASCUAL FONTANILLES (Universitat Rovira i Virgili)

Degree
Master's Degree in Artificial Intelligence

Master's thesis

School of Engineering
Universitat Rovira i Virgili (URV)

Faculty of Mathematics
Universitat de Barcelona (UB)

Barcelona School of Informatics (FIB)
Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

11/05/2026

Abstract

Contemporary LLM-based orchestration platforms such as LangGraph and CrewAI enable rapid composition of agent pipelines, but share three architectural weaknesses. They lack a persistent record of peer performance across runs, expose no formal model of agent capabilities and task constraints, and treat compliance and ethics as ad hoc concerns rather than first-class design primitives. Consequently, allocation quality does not improve with experience and pipelines cannot demonstrate structural or regulatory correctness, a limitation made urgent by emerging regulatory regimes such as the EU AI Act and the NIST AI Risk Management Framework, which codify binding and de facto expectations for transparency and human oversight in high-risk AI deployments.

This thesis presents a stigmergy-driven Multi-Agent framework for intelligent task orchestration that addresses all three limitations within a single integrated design. Coordination is mediated by digital pheromone markers deposited on a shared semantic knowledge graph, where a Pheromone Engine reinforces successful assignment patterns while allowing stale ones to decay, so that experience accumulates in the environment rather than in any individual agent and emergent optimisation arises without direct inter-agent communication. Task allocation builds on an Improved Contract Net Protocol whose bid scoring is augmented with this accumulated pheromone history. Structural integrity is enforced by a semantic validation layer, while an Ethics-Check Agent provides a domain-agnostic safety gate that screens all pipeline outputs for hallucinated data, unfair bias, sensitive-data leakage, and unsafe approvals before execution proceeds.

An extensive controlled evaluation in a commercial warehouse brokerage scenario supports all five hypotheses. The pheromone engine substantially improves task allocation accuracy as experience accumulates, the dual-layer governance achieves near-complete recall of ground-truth violations at a low false-positive rate, and the full framework outperforms an uncoordinated baseline by a substantial margin at modest latency and token overhead. A user perception survey with domain experts and technical evaluators indicates that the system is regarded as suitable for internal first-draft generation, with mandatory human review required before external use.

This work demonstrates how swarm intelligence, the Semantic Web, and modern AI orchestration can be fused to create a Multi-Agent platform that is adaptive, auditable, and governance-aware by design. The full implementation is publicly available at `https://github.com/evaveli/agentcy`.

Acknowledgments

I would like to express my sincere gratitude to my supervisor for their guidance, insight, and support throughout the development of this thesis. Their feedback has shaped this work at every stage and helped me grow considerably as a researcher.

I would also like to thank Estoko Logistics, the company where I work, for their invaluable contribution to this thesis. Their support in validating the logistics scenarios, establishing the ground truths, and participating in the user perception survey was essential to the empirical evaluation of this work, and this thesis would not have been possible in its current form without their collaboration.

I am deeply thankful to all the friends I met during the Master's Degree in Artificial Intelligence programme, whose companionship made this journey not only manageable but genuinely enjoyable. A special thank you goes to Tatevik Davtyan and Skaise Butkute, with whom I shared countless sleepless nights of work, the shared stress of exams and assignments, and the many study sessions that turned the hardest moments of this master into some of the most meaningful. Thank you for becoming my family when my real one was 2,125 kilometres away. Our time together has become a part of my everyday life that I know will remain with me long after this master ends.

To my partner, thank you for your unwavering support and for everything you did for me during the long stretches when I could not leave my room because of exams and assignments. Your patience, care, and presence through those difficult moments meant more than I can express.

To my family, thank you for your unconditional love, patience, and encouragement throughout this journey. You created a life in which I have had the opportunity to pursue my dreams, receive an education, and see the world. None of this would have been possible without you.

Finally, I would like to acknowledge myself for the perseverance, discipline, and belief that carried me through every challenge along the way.

Table of Contents

List of Figures

List of Tables

List of Terms

ACC Assignment Accuracy, the ratio of correct agent assignments to total assignments, defined in Table 12 and used as the primary quality metric throughout the ablation study.

ACID Atomicity, Consistency, Isolation, Durability, a set of properties that guarantee reliable processing of database transactions.

AffordanceMarker `stig:AffordanceMarker`, the ontology class representing a stigmergic marker that records an agent-task pairing together with its bid score and accumulated intensity.

AMQP Advanced Message Queuing Protocol, an open standard application layer protocol for message oriented middleware, used here by RabbitMQ.

API Application Programming Interface, a defined set of endpoints and contracts through which software components communicate.

C0–C7 Ablation Configurations 0 to 7, the eight ablation configurations defined in Subsection 6.2.1, where C0 denotes the full framework and C7 the minimal coordination-free baseline.

CFP Call for Proposals, the initial broadcast message in the Contract Net Protocol that invites agents to bid on a task.

CNP Contract Net Protocol, a negotiation-based task-allocation mechanism in which a manager broadcasts a call for proposals and selects the best bid [1].

CO_2e Carbon Dioxide Equivalent, the unit used to express the global warming potential of greenhouse gas emissions, applied in the environmental footprint analysis of Section 7.

CRUD Create, Read, Update, Delete, the four basic operations exposed by the Agent Registry Store.

DAG Directed Acyclic Graph, a graph with directed edges and no cycles, used here to represent task-execution ordering.

DAI Distributed Artificial Intelligence, the subfield of AI studying how multiple loosely coupled reasoning entities can solve problems collectively.

DG1–DG5 Design Goals 1 to 5, the five framework design goals defined in Section 4.1.

EU AI Act Regulation (EU) 2024/1689, the European Union legal framework governing high-risk artificial intelligence systems [2].

F1 F1 score, the harmonic mean of precision and recall, used to evaluate the Compliance Agent as a binary classifier over client-warehouse pairings.

FCFS First-Come, First-Served, an assignment policy where the first agent to claim a task is awarded the slot.

FIPA Foundation for Intelligent Physical Agents, a standards body that publishes agent communication specifications.

GDPR General Data Protection Regulation, the European Union regulation governing the processing of personal data [3].

GPT Generative Pre-trained Transformer, the family of transformer based language models from OpenAI from which GPT-4o-mini, used throughout the experimental evaluation, is drawn [4].

H1–H5 Hypotheses 1 to 5, the five evaluation hypotheses defined at the start of Chapter 6 and synthesised in Table 4.

HIL Human-in-the-Loop, a design pattern in which a human operator reviews, approves, or overrides automated decisions.

HTTP HyperText Transfer Protocol, the application protocol underlying the FastAPI endpoints, the SPARQL Graph Store Protocol, and the OpenTelemetry trace propagation used in the prototype.

ICNP Improved Contract Net Protocol, an enhanced variant of the CNP that incorporates stigmergic markers and pheromone-weighted scoring [5].

IE Interaction Effect, the factorial term $\mathrm{ACC}(C0) - \mathrm{ACC}(C3) - \mathrm{ACC}(C4) + \mathrm{ACC}(C3{+}C4)$ used to test super-additive interaction between the two validation layers.

IQR Interquartile Range, the spread between the 25th and 75th percentiles of a distribution.

IRI Internationalized Resource Identifier, a generalisation of the URI that supports a wider character set, used as a global identifier in Linked Data [6].

JSON JavaScript Object Notation, a lightweight text-based data interchange format used for task payloads and audit records.

LAT End-to-End Latency, the elapsed time from pipeline start to final audit log timestamp, defined in Table 12.

LLM Large Language Model, a neural network trained on large text corpora, used for natural language understanding and generation.

MAS Multi-Agent System, a system composed of multiple interacting autonomous agents.

N1QL Couchbase query language for JSON documents, used for cross-collection audit queries [7].

NIST AI RMF National Institute of Standards and Technology AI Risk Management Framework, a voluntary governance framework for identifying and mitigating risks in AI systems [8].

O1–O6 Objectives 1 to 6, the six thesis objectives stated in Chapter 1 and operationalised through the five design goals.

OTLP OpenTelemetry Protocol, a vendor-neutral wire protocol for exporting traces, metrics, and logs from instrumented services.

OWL Web Ontology Language, a W3C standard for defining formal ontologies on the Semantic Web [9].

PII Personally Identifiable Information, sensitive content that is excluded from structured audit payloads to preserve participant privacy.

pp Percentage points, the absolute difference between two percentage values, distinguished from a relative percentage change.

PROV-O W3C Provenance Ontology, a formal vocabulary of entities, activities, and agents for representing and exchanging provenance information [10].

RDF Resource Description Framework, a W3C standard data model for representing information as subject, predicate, object triples [11].

RDFS RDF Schema, the W3C vocabulary for describing classes and properties of RDF resources.

ReservationMarker `stig:ReservationMarker`, the ontology class that prevents an assigned agent from being double-booked by concurrent pipelines.

REST Representational State Transfer, an architectural style for stateless HTTP-based web services.

RQ Research Question, the central question guiding the thesis, defined in Chapter 1.

RSR Recovery Success Rate, the proportion of injected failures successfully resolved by the recovery pipeline, defined in Table 12.

RT Recovery Time, the elapsed time from an injected failure to pipeline completion, with a 300 s timeout, defined in Table 13 and reported in Table 30.

SDG Sustainable Development Goals, the seventeen United Nations goals referenced in Section 7.5 [12].

SDK Software Development Kit, a collection of libraries and tools provided by a vendor to integrate with their service.

SHA-256 Secure Hash Algorithm with 256-bit digest, used here to fingerprint cached plan drafts.

SHACL Shapes Constraint Language, a W3C standard for validating RDF graphs against a set of conditions [13].

SIEM Security Information and Event Management, a class of tools that aggregate, correlate, and analyse audit and security events.

SKOS Simple Knowledge Organization System, a W3C recommendation for representing thesauri, taxonomies, and controlled vocabularies.

SPARQL SPARQL Protocol and RDF Query Language, a query language for retrieving and manipulating data stored in RDF format [14].

Stigmergy A mechanism of indirect coordination in which agents communicate by modifying a shared environment, for example by depositing pheromone markers, rather than through direct messaging [15].

TaskSpec `stig:TaskSpec`, the ontology class representing a normalized, structured task specification in the proposed framework.

TOK Token Consumption, the sum of LLM input and output tokens per pipeline run, defined in Table 12.

TPR / FPR True Positive Rate and False Positive Rate, standard classification metrics used to evaluate the validation layers.

TTL Time-To-Live, the validity window within which agents may submit bids in response to a CFP.

URI Uniform Resource Identifier, a compact string that unambiguously identifies an agent, capability, or task.

UTC Coordinated Universal Time, the reference time standard used for all audit log timestamps.

VR Constraint Violation Rate, the proportion of runs containing at least one BLOCK violation from plan validation or compliance output, defined in Table 12.

1. Introduction

The rapid expansion of artificial intelligence into enterprise operations has shifted the task-management problem from single-model pipelines to populations of specialised autonomous agents. Distributed AI systems are increasingly required to decompose complex objectives, allocate subtasks to heterogeneous agents with non-overlapping capabilities, and enforce domain-specific constraints. They must also adapt their behaviour over time without delegating authority to a monolithic central controller, and with full auditability of every decision. Multi-Agent Systems (MAS) [16] provide the natural computational substrate for this kind of collaborative decomposition, and the recent integration of Large Language Models into agent architectures [17] has opened new possibilities for flexible, natural language driven coordination. Yet the deployment of LLM-based MAS in production environments exposes a set of recurring architectural limitations that motivate the work presented in this thesis.

Modern orchestration frameworks, including LangGraph [18], CrewAI [19], AutoGen [20], and others, have substantially lowered the barrier to composing chains of LLM-powered agents. Although these frameworks differ in abstraction level and coordination philosophy, and several offer meaningful within-run or session-level persistence such as LangGraph's checkpointed state and CrewAI's long-term memory layers, they share a deeper architectural limitation. Coordination is stateless with respect to agent performance history across pipeline runs. No framework maintains a structured record of which agents have historically performed well on which task types, nor does any expose a formal ontological model of capabilities and constraints, or a principled mechanism for encoding and enforcing domain-specific rules (e.g. regulatory constraints) and cross-cutting safety concerns (e.g. hallucinated data, unfair bias, sensitive-data leakage, and unsafe approvals). The consequence is a system that can execute a workflow but cannot improve its allocation decisions over successive runs, and that offers no structural or regulatory correctness guarantees. LangGraph and CrewAI are adopted throughout this thesis as representative case studies because they occupy complementary positions in the current design space, with LangGraph exemplifying graph-centric stateful control and CrewAI embodying role-based collaboration, and together they expose the full spectrum of limitations common to the broader class of orchestration platforms. A detailed justification of this selection is provided in Chapter 3.

The urgency of addressing these limitations is compounded by converging pressures from industry and regulation. Enterprise adoption of autonomous AI agents is constrained by legitimate concerns about auditability, regulatory compliance, and operational risk. The EU AI Act (Regulation EU 2024/1689) [2] establishes binding legal requirements for transparency, contestability, and human oversight in high-risk AI deployments. The NIST AI Risk Management Framework [8] reinforces these expectations as a voluntary but widely adopted governance standard.

An orchestration framework that cannot explain its allocation decisions, enforce domain constraints by design, or demonstrate that its behaviour improves predictably over time will not meet these standards. The gap between current framework capabilities and regulatory expectations therefore constitutes not merely a theoretical shortcoming but a concrete barrier to responsible industrial deployment. The specific regulatory requirements and the architectural elements of the proposed framework that address them are discussed in detail in Chapters 4 and 7.

Three interdependent architectural limitations emerge from this analysis. Crucially,

these limitations are rooted in the core design of existing orchestration frameworks rather than in specific implementation choices. No amount of plugin development or configuration can supply coordination primitives that the underlying architecture omits.

First, existing frameworks are architecturally stateless. They treat each pipeline execution as independent, discarding accumulated performance evidence after every run. Swarm intelligence research has long demonstrated that stigmergic coordination, that is, indirect communication through shared environmental markers, enables emergent, adaptive collective behaviour [15], a property that arises because agents influence one another solely through the environment rather than through direct peer-to-peer messaging. Applying stigmergy to LLM-based MAS [17, 21] in a formally grounded way has not been systematically explored.

Second, the architectures of current frameworks provide no formal capability model. Agent capabilities, task requirements, and domain constraints are instead encoded as unstructured prompts or opaque JSON schemas, precluding automated constraint checking, semantic interoperability, and provenance tracking. The Semantic Web [22] stack offers a machine-interpretable alternative validated in domains requiring explainability and auditability, with OWL and RDF for ontological modelling, SHACL for structural constraint validation, and SPARQL for querying [22].

Third, existing frameworks provide no first-class architectural primitives for compliance checking or ethics screening. Governance is addressed only through ad hoc logging rather than as a structural property of the system.

Taken together, these three gaps define the problem space that this work addresses. The proposed **Stigmergy-Driven Multi-Agent Framework for Intelligent Task Orchestration** is a domain-agnostic architecture that resolves all three limitations within a single integrated design. The framework introduces a stigmergic coordination medium [15] in which agents post stig:AffordanceMarkers and stig:ReservationMarkers, which serve as digital pheromones, to a shared OWL/RDF knowledge graph. A Pheromone Engine continuously decays and reinforces marker intensities so that successful assignment patterns accumulate over time, enabling the system to improve its allocation quality across successive pipeline runs. Task allocation is managed through Improved Contract Net Protocol (ICNP) [5], which augments the classical Contract Net Protocol [1] bid scoring with pheromone-weighted history, producing a self-correcting market mechanism. Structural integrity of generated plans is enforced by a SHACL validation layer, an Ethics-Check Gate screens agent outputs against safety and fairness criteria [8, 23] before execution, and the architecture exposes a pluggable validation hook through which domain-specific business rules can be enforced at a complementary abstraction level, enabling a defence-in-depth governance pattern. The entire framework is deployed as a suite of cloud-native microservices communicating via an Advanced Message Queuing Protocol message broker and persisting state in a document store, with per-run isolation, idempotent execution, and structured audit logging built into the infrastructure from the outset. The coordination mechanisms are expressed over abstract ontological primitives and carry no domain-specific semantics. Domain knowledge resides exclusively in pluggable agent implementations and SHACL constraint shapes. The full prototype is released as an open-source artefact at `https://github.com/evaveli/agentcy`.

The thesis is motivated by the following central research question.

> **RQ.** To what extent can the integration of stigmergic coordination, formal ontological modelling, and layered governance mechanisms into an LLM-based Multi-Agent framework overcome the stateless coordination of agent performance history, the absence of a formal capability model, and the lack of compliance enforcement that are inherent in current orchestration platforms?

This question is addressed through both design and empirical evaluation. The design contribution (Chapter 4) proposes an integrated architecture that fuses stigmergic pheromone markers, an Improved Contract Net Protocol with pheromone-weighted bidding, an OWL/RDF ontological substrate, SHACL structural validation, an Ethics-Check Gate, and a multi-tier failure recovery stack. The empirical contribution (Chapter 6) tests whether each coordination mechanism delivers its predicted benefit through five hypotheses operationalised across six complementary experiments. These experiments comprise a 790-run ablation study spanning nine framework configurations, a pheromone convergence experiment tracking adaptive learning over successive runs, a priority-based estimator selection analysis, a validation layer interaction analysis examining the complementarity of SHACL and domain-specific checks, a failure recovery evaluation measuring tiered self-healing behaviour, and a user perception survey with $N = 19$ domain expert and technical evaluators. The framework is instantiated and evaluated in a commercial warehouse brokerage scenario, chosen because it exhibits the coordination challenges (heterogeneous agent capabilities, dynamic re-planning, regulatory constraints, and consequential decisions) that are representative of the broader class of problems the framework targets.

The thesis pursues six objectives.

- **O1.** To design a stigmergic coordination mechanism that enables LLM-based agents to learn collectively from accumulated task-allocation history without requiring direct inter-agent communication.
- **O2.** To integrate the Improved Contract Net Protocol (ICNP) with pheromone-weighted bid scoring within the stigmergic coordination medium, producing adaptive, history-informed task allocation.
- **O3.** To construct a formal OWL/RDF ontology and SPARQL audit infrastructure that provides a machine-interpretable, provenance-preserving knowledge substrate for a Multi-Agent task pipeline.
- **O4.** To design a governance mechanism that enforces structural constraints and domain-agnostic safety guarantees by design, while providing an extensible pipeline for domain-specific validation agents, achieving governance-by-default rather than governance by afterthought.
- **O5.** To design a multi-tier failure recovery architecture that combines bounded retries, LLM-driven strategic replanning, and pheromone-based re-weighting to achieve adaptive resilience without mandatory human intervention.
- **O6.** To implement and empirically evaluate a cloud-native prototype of the full framework, quantifying the individual and combined contribution of each coordination mechanism to system quality and measuring the associated overhead costs.

These six objectives are formalised as the five design goals of the framework in Section 4.1, which also specifies how each goal maps to the experimental hypotheses evaluated in Chapter 6.

The principal contributions of this thesis are as follows. **(1)** A novel stigmergic coordination architecture for LLM-based Multi-Agent Systems, in which agents negotiate task assignments indirectly through AffordanceMarkers and ReservationMarkers posted to a shared knowledge graph, enabling emergent adaptive optimisation. **(2)** A pheromone-weighted ICNP in which bid scoring incorporates accumulated historical markers, producing a self-improving allocation mechanism. Together, these two mechanisms yield an accuracy gain from approximately 10% to 96% over 14 convergence runs ($\Delta = +86$ percentage points). **(3)** A formal OWL/RDF ontology covering the full task-pipeline lifecycle, encompassing submission, allocation, validation, execution, and outcome recording, together with SPARQL, queryable audit infrastructure for provenance and traceability. **(4)** A governance architecture combining SHACL structural validation with a domain-

agnostic Ethics-Check Gate, and an extensible pipeline that supports domain-specific validation agents, providing defence-in-depth coverage across complementary failure classes. Empirically, SHACL and the Compliance Agent exhibit zero detection overlap (6/6 structural defects caught by SHACL alone, 3/3 domain defects caught by the Compliance Agent alone), and the Compliance Agent achieves TPR = 1.000, FPR = 0.011, and F1 = 0.992 as a standalone classifier over 300 evaluated client-warehouse pairings. **(5)** A fully operational cloud-native prototype evaluated across 790 controlled pipeline runs, demonstrating a +48 percentage point accuracy advantage of the full framework over a coordination-free baseline at a measured cost of only +6.6% additional latency and +9.7% additional token consumption.

The remainder of this thesis is organised as follows. Chapter 2 introduces the conceptual and technical background, covering Multi-Agent Systems and coordination theory, the Contract Net Protocol, LLM-based agents with tools and memory, the Semantic Web stack, stigmergy, cloud-native orchestration patterns, and representative task-assignment policies. Chapter 3 surveys the state of the art, providing an in-depth analysis of LangGraph and CrewAI and mapping their limitations against the three gaps addressed in this work. Chapter 4 presents the proposed framework in full, covering design goals, agent registration, task normalisation, the ICNP bidding mechanism, plan generation, SHACL validation, DAG construction, human-in-the-loop routing, the Pheromone Engine, the LLM Strategist, the Ethics-Check Gate, failure handling, and audit traceability. Chapter 5 describes the cloud-native prototype implementation. Chapter 6 reports the experimental evaluation, covering the hypotheses, experimental setup, quantitative ablation and convergence results, statistical analysis, and a qualitative user perception study. Chapter 7 analyses sustainability and ethical implications. Chapter 8 concludes the thesis and outlines directions for future research.

2. Background

This chapter introduces the conceptual and technical foundations that this work builds on. It begins with classical Multi-Agent Systems (MAS) [16], outlining their core properties, historical development, and coordination mechanisms, including the Contract Net Protocol [1]. It then turns to LLM-based Multi-Agent Systems [17], where large language models act as agents equipped with tools, memory, and human-in-the-loop oversight. Next, key ideas from the Semantic Web [22] that provide a shared, machine-interpretable substrate for representing knowledge, constraints, and provenance. Stigmergy [15] is also discussed as a principle of indirect, environment-mediated coordination, related to distributed, self-organising behaviour. Building on this, cloud-native orchestration patterns are summarised, such as message-driven microservices, DAG-based workflows, fault tolerance, and transactional consistency, that inform scalable agent execution. Finally, the representative assignment policies used to allocate tasks to agents are reviewed, ranging from simple First-Come, First-Served rules to improved Contract Net variants and utility-based scoring schemes.

2.1 Multi-Agent Systems

A Multi-Agent System is a system composed of multiple intelligent agents (either software or hardware entities) that interact to collectively solve problems, which would be difficult or impossible for a single agent or a purely centralised system to solve [16, 24]. Within a MAS, each agent operates autonomously, perceiving its environment and making decisions based on its own goals and rules. Unlike a traditional system with one central controller directing every action, a MAS consists of many semi-independent agents working within a common environment. They may collaborate or compete with one another, but together they aim to achieve overarching system goals. In essence, the power of a MAS comes from the decentralization of intelligence. Instead of one entity handling everything, multiple entities contribute their specialised capabilities.

To illustrate these principles in practice, Figure 1 presents a typical MAS architecture. It includes a central message bus (based on a publish/subscribe model [25]) that mediates communication among autonomous agents. These agents follow a Perception $\rightarrow$ Reasoning $\rightarrow$ Action loop [26] and share situational state via a shared world model or blackboard [27]. They persist learned or observed knowledge in a knowledge base, interact with an environment or simulator, integrate with external services, and generate monitoring and logging data. The architecture also allows for interaction with an operator dashboard and optionally a coordinator, supporting decoupled coordination, scalable messaging, and shared situational awareness.

Key characteristics of Multi-Agent Systems include [24]:

- **Autonomy**: Each agent acts independently without direct external control. Agents make decisions based on their own state, perceived environment, and predefined objectives.
- **Decentralization**: There is no single point of control, decision-making is distributed among agents. This peer-to-peer architecture avoids a central bottleneck and improves fault tolerance.

- **Interaction and Coordination**: Agents communicate and coordinate their actions to work towards common goals. Effective communication protocols allow the team of agents to solve problems collaboratively, often more efficiently than isolated actors.
- **Emergent behaviour**: Complex, higher-level behaviours or solutions can emerge from the local interactions of many simple agents. The collective intelligence of the group can solve problems in dynamic ways that no single agent was explicitly programmed to handle.

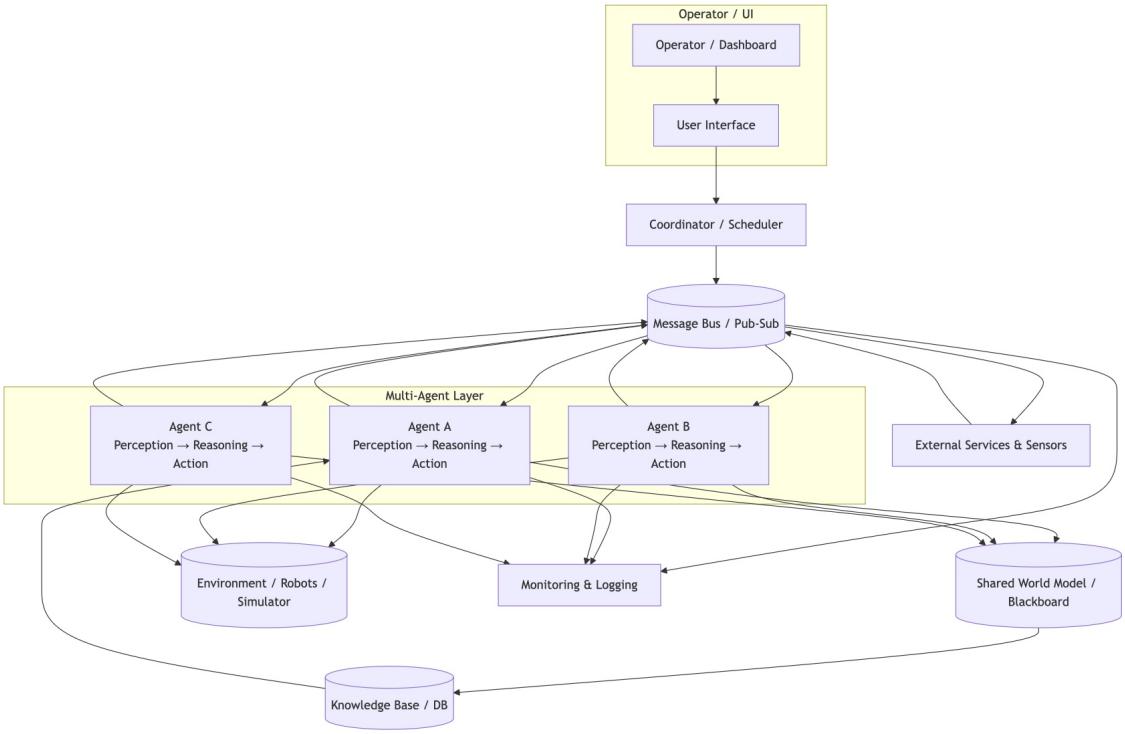


Figure 1: Architectural overview of a Multi-Agent System.

By distributing intelligence and work among multiple agents, MAS offer significant advantages over traditional single-agent systems. They are more scalable and adaptable. A MAS can handle growing complexity by simply adding more agents, or by having agents dynamically adjust their strategies. They are also inherently more robust. If one agent fails or is suboptimal, others can compensate, preventing a total system failure. In fact, Multi-Agent architectures have demonstrated robust adaptability, fault tolerance, and the capacity for solving complex problems in dynamic settings far beyond what a single, monolithic system could achieve [24]. Furthermore, MAS are modular by design, each agent can be specialised for a particular task or domain. This modularity makes it easier to integrate heterogeneous components or legacy systems by assigning interface agents to each component. Overall, the collaborative and decentralised nature of MAS directly addresses the bottlenecks of traditional systems, instead of one agent trying to do everything, many agents can divide and conquer the problem space.

2.1.1 History of Multi-Agent Systems

The concept of Multi-Agent Systems is not new. It traces back to early research in Distributed Artificial Intelligence (DAI) [28] in the 1970s and 1980s. Researchers realised

that framing complex, distributed problems as interactions among multiple autonomous "agents" (rather than one giant program) could yield more scalable and resilient solutions. Early work in the late 1970s explored distributed problem-solving networks and blackboard systems, where independent programs communicated through a common data space. In 1980, Reid G. Smith introduced the Contract Net Protocol [1] for decentralised task allocation among agents (more in Subsection 2.1.3). Through the 1980s, foundational theories for MAS communication and cooperation were developed, including formal agent communication languages and negotiation mechanisms for conflict resolution [24].

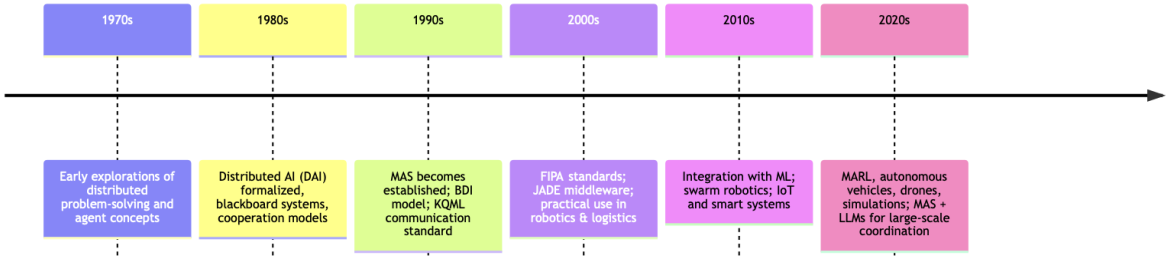


Figure 2: Timeline of MAS evolution from the 1970s to the 2020s.

By the 1990s, MAS had become an established research area. Notable milestones included the Belief–Desire–Intention (BDI) agent architecture [29] (formalising an agent's mental state and decision process), and standardised frameworks such as the Knowledge Query Manipulation Language (KQML) [30] and later the FIPA Agent Communication Language (ACL) [31]. These provided common protocols for agents to exchange messages and understand each other's queries or assertions. Dedicated conferences and journals emerged, bringing MAS into the AI mainstream. In the late 1990s and early 2000s, toolkits like JADE [32] (Java Agent DEvelopment Framework) provided practical platforms to implement MAS, and the FIPA standards organisation published interoperability specs, enabling Multi-Agent platforms to share common communication and ontology definitions.

MAS research and applications continued to expand through the 2000s and 2010s. Multi-Agent Systems were applied in robotics (teams of robots/drones coordinating), distributed sensing (sensor networks), logistics and manufacturing, simulation of social or economic systems, and more [24]. Advances in machine learning also influenced MAS, for example, Multi-Agent reinforcement learning [33] allowed agents to learn optimal behaviours through trial-and-error in shared environments. By the 2010s, MAS techniques were used in complex domains like smart grids (agents controlling distributed energy resources), traffic control, and collaborative multi-robot exploration. Figure 2 illustrates a rough timeline of MAS evolution from the 1970s to today, highlighting these key developments.

Today, interest in MAS is resurging thanks to new possibilities with AI agents (discussed in Section 2.2). Modern MAS implementations range from swarms of simple devices to sophisticated ensembles of AI services in enterprises. The continued importance of MAS lies in their ability to handle the complexity and interconnectedness of real-world problems. By breaking a problem into parts and assigning them to diverse agents, MAS avoid the brittleness of monolithic approaches.

2.1.2 Coordination in Multi-Agent Systems

Because a MAS involves multiple autonomous agents, a central challenge is coordination, which concerns how to ensure agents work together coherently toward system goals. Agents must share information, divide tasks, and avoid interfering with each other. Broadly, coordination in MAS can be achieved through two modes: direct communication and indirect coordination [24].

In direct communication, agents send messages to one another (peer-to-peer or via an arbitrator) to negotiate and synchronise [24]. For example, one agent might request help from another, or they might exchange state updates. This requires a common language/protocol (such as FIPA ACL messages or a custom schema) so that agents interpret messages correctly. Direct communication is akin to how a team of humans might talk or email to coordinate tasks.

In indirect coordination, agents coordinate by acting on a shared environment or data repository that all can observe. This is often called a blackboard [27]. Instead of sending Agent B a message, Agent A might post an "alert" or leave a marker in the environment that any interested agent can detect and respond to. This decouples agents in time and space. They need not be connected, nor know each other's identities, provided they can read from and write to the shared medium.

Many MAS use a combination of these approaches. A common pattern is a shared message bus or publish/subscribe system [25], where agents publish events or task announcements to a broker and other agents subscribed to those topics receive them. This behaves like a digital blackboard where messages are the chalk marks. Another pattern is an explicit coordinator agent that collects inputs and assigns tasks (a bit more centralised, sometimes called orchestrator). Alternatively, fully distributed protocols allow agents to reach consensus or contracts without any central authority [24] (e.g. via voting or distributed constraint solving).

2.1.3 Contract Net Protocol (CNP)

One of the earliest and most influential MAS coordination mechanisms is the Contract Net Protocol (CNP) [1], introduced by Smith. CNP is a decentralised protocol for dynamically assigning tasks to agents through a negotiation process.

It follows an auction-like call for bids model, often explained in terms of managers and contractors.

- **Call for Proposals (CFP):** As depicted in step (a) of Figure 3, the manager agent broadcasts a task announcement to potential contractors. The CFP specifies the task requirements and any bid constraints such as deadlines or capability prerequisites.

- **Bid submission:** In step (b), contractors evaluate the CFP. Those willing and able to perform the task respond with a proposal, while others issue a refusal (also shown in Figure 3). Proposals typically include estimates such as expected cost, duration, or quality.

- **Awarding the contract:** After the bidding deadline, the manager compares the received proposals, represented by the branching logic in step (c) of Figure 3. It selects the most suitable proposal and sends an acceptance message to the chosen contractor, and sends rejection messages to the remaining bidders.

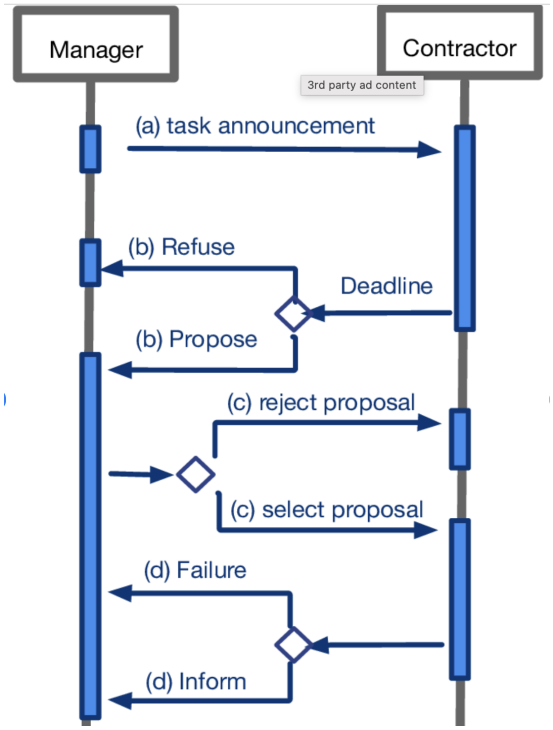


Figure 3: The Contract Net Protocol sequence diagram (reproduced from [34]).

- **Execution and confirmation:** Finally, as shown in Figure 3(d), the selected contractor carries out the task and returns either a success inform message or a failure notification, closing the contract instance.

CNP is a systematic way to achieve distributed task allocation. Rather than a central scheduler assigning tasks blindly, CNP lets the agents themselves signal their own suitability or availability for a task by bidding. This is useful in systems where agents have different capabilities or loads, the bidding process yields a natural selection of the most appropriate agent for each job. It is analogous to a subcontracting process in human teams or an auction in a marketplace. Importantly, CNP is a peer-to-peer protocol. This means that any agent can play the role of manager for tasks it needs done, and agents can be contractors for each other. This supports a very flexible, decentralised workflow.

However, CNP introduces some communication overhead. It requires multiple message exchanges (CFP, potentially multiple bids, then award and confirmations). In settings with many agents or tasks, a naive implementation could flood the network with CFPs and bids. Smith's original paper discussed strategies to mitigate this, for example, limit the CFP to relevant agents only, keep messages short, or skip the bidding if the manager already knows a suitable contractor. Modern MAS often use such optimisations.

2.2 LLM-Based Multi-Agent Systems

In recent years, Large Language Models (LLMs), such as GPT-style transformers capable of understanding and generating human-like text, have fundamentally changed the design of Multi-Agent Systems. LLM-based Multi-Agent Systems refer to MAS in which one or more agents are powered by large language models [17]. This represents a convergence of natural language AI with Multi-Agent architectures, enabling novel forms of collaboration and problem-solving. LLMs have shifted from passive text generators to autonomous agents that perceive, reason, plan, and act, all through the medium of language [35].

One of the biggest changes in LLM-based MAS is how agents communicate and reason. Traditional MAS often used fixed protocols or structured messages for agent communication (as discussed in Subsection 2.1.2). In an LLM-based MAS, agents can use natural language to interact. This means agents can flexibly exchange information, ask each other questions, explain their reasoning, and coordinate plans in a very human-like conversational manner. Such rich language-based coordination enables unprecedented flexibility, agents can negotiate roles or adjust strategies on the fly by literally "talking it out" in English (or another language), which is something rigid APIs struggle with. Early experiments have demonstrated emergent behaviours, where two LLM agents adopting the roles of developer and evaluator can iteratively refine code solutions through dialogue, mirroring a human pair programming session [36]. The collective intelligence emerging from multiple LLM agents working together can surpass what a single model could achieve alone [37].

This trend gained rapid momentum around 2023. Projects like CAMEL (Communicative Agents for "Mind" Exploration of LLM Society) demonstrated that two ChatGPT-based agents could brainstorm together to solve tasks, with one agent taking the user role and another the assistant role, refining solutions through dialog [37]. Other work explored "self-reflection" [38] where an LLM splits into a reasoner and a checker to improve its answers. By 2024, frameworks such as Microsoft's Jarvis (HuggingGPT) [39, 40] and Significant-Gravitas's AutoGPT [41] and others popularised the idea of an LLM orchestrating other specialist models or tools, effectively an agent society. Researchers have formalised these into Multi-Agent simulations (e.g. generative agents that simulate communities of LLM characters [42]) and Multi-Agent collaboration setups for tasks like coding, where different agents take on roles like Writer, Tester, Debugger. The industry is also embracing this shift, with enterprise AI solutions moving away from a single monolithic chatbot toward collections of specialised LLM agents coordinated by an orchestrator, analogous to a human team managed by a designated leader [43]. One agent might focus on retrieving information, another on performing calculations or API calls, another on composing final reports, etc., with a central agent delegating and integrating their outputs. This design avoids the pitfalls of trying to have one giant model do everything, and leverages the principle of "division of labour" for AI.

2.2.1 Agents, Tools, Memory, and Human-in-the-Loop

In an LLM-based MAS, each agent is typically built around an LLM "brain" augmented with additional capabilities. A single LLM on its own is limited by context window and lack of persistent memory or real-world actuation, so recent agent frameworks extend them with tools and memory [35]:

- **Tools -** LLM agents can be equipped with external tools such as search engines, calculators, databases, or APIs for specific functions. For example, an LLM agent may call a web search API to gather information, use a code execution sandbox to run computations, or query a knowledge base. Incorporating tools allows agents to affect the environment and get up-to-date information beyond their training data.

- **Memory -** Large language models have a context length limitation, they cannot remember arbitrarily long histories of interaction. To mitigate this, LLM agents are given auxiliary memory systems [42]. This might be a vector database of relevant facts they can retrieve, a long-term knowledge store of what has happened in the conversation/task so far, or scratchpad notes. By storing and recalling past information, agents avoid repeating mistakes and maintain consistency. For instance,

an agent can recall earlier subtasks completed or decisions made, even if that was thousands of tokens ago, by retrieving from an external memory and feeding it back into the prompt.

- **Autonomy with safe boundaries -** Each LLM agent runs autonomously in its role, but a human-in-the-loop (HITL) is retained for oversight when needed. Human oversight can be invoked in critical scenarios, for example, if agents reach a decision that violates certain constraints or if they deadlock. Designing hooks for HITL is important for safety [44]. In practice, this could mean a human supervisor gets notified and can intervene (approve, modify, or cancel actions) or simply that the system keeps a human-readable log for audit. Many critical applications (finance, healthcare) mandate human oversight on AI decisions.

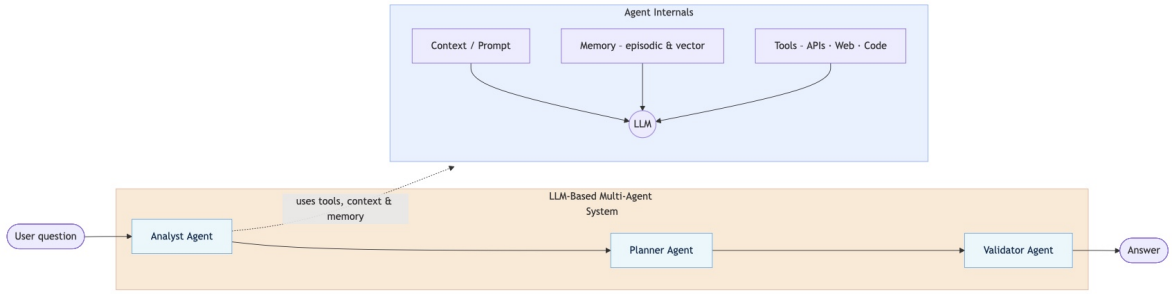


Figure 4: Diagram of an LLM Multi-Agent System where agents consult prompt context, memory, and tools to transform a user question into an answer.

In sum, wrapping an LLM with tools, external memory, and human-in-the-loop safeguards turns a stateless text model into a dependable, task-conditioned service. Tools let agents observe and act on the world, memory supplies durable context beyond the token window and HITL checkpoints provide oversight, escalation, and auditability. Together with the structured prompt context shown in Figure 4, these elements align behaviour with goals and constraints, reduce mistakes and drift, enable safe retries and recovery, and make agent decisions reproducible and explainable within the larger MAS.

2.3 Semantic Web

The Semantic Web [22] corresponds to a stream of work in AI and Web technologies that aims to make data on the web and within organisations, not only human readable, but machine interpretable through explicit, shared semantics. Instead of treating information as isolated documents or opaque JSON structures, the Semantic Web stack models knowledge as a labelled graph of entities and relations, expressed primarily through the Resource Description Framework (RDF) [11], where statements are encoded as subject–predicate–object triples. On top of this, schema and ontology languages such as RDFS and OWL [9] enable the definition of controlled vocabularies, class hierarchies, and logical constraints, providing a formally grounded way to describe domain concepts (e.g. tasks, agents, resources, regulations, datasets) and their interdependencies. This logical layer is not merely descriptive. It supports automated reasoning (e.g. subsumption, consistency checking, classification), typically based on Description Logics [45] and the open world assumption, allowing systems to infer implicit facts from explicitly asserted ones and to validate whether data and behaviours comply with modeled constraints. SPARQL [14]

offers a standard query language for retrieving and integrating this distributed graph-structured knowledge, while Linked Data principles [46] and the use of Internationalized Resource Identifiers (IRIs) [6] as global identifiers make it possible to interconnect independently maintained datasets into larger, interoperable knowledge graphs. Over time, this ecosystem has been extended with complementary standards and profiles (such as SKOS [47] for concept schemes, PROV-O [10] for provenance, and SHACL [13] for constraint validation), and has been widely adopted in domains like life sciences, digital libraries, geospatial systems, enterprise integration, and policy/contract modelling, where explainability, interoperability, and long-term maintainability are critical [22].

PROV-O (the W3C Provenance Ontology) provides a formal vocabulary of entities, activities, and agents for representing and exchanging provenance (that is, the origins, context, and lineage of data or artefacts across systems), enabling consistent capture of how data was produced or transformed and supporting reasoning about trust and reproducibility [10]. SHACL (Shapes Constraint Language) is a W3C recommendation for defining and validating structural constraints over RDF graphs, where ontology engineers specify expected shapes that are expressed in RDF and used to automatically check conformance, ensuring data integrity and consistency with the ontology [13]. By providing a formal mechanism to impose closed world style constraints on an open-world data model, SHACL complements the expressive power of RDF/OWL with a practical means of data validation.

For autonomous systems and Multi-Agent settings in particular, the Semantic Web provides a common semantic substrate in which environment state, roles, objectives, norms, and interaction protocols can be represented in a uniform, technology agnostic manner. Heterogeneous agents can interpret shared structures consistently, discover relevant resources or services via their semantic descriptions, and align their local decision-making with global constraints encoded in ontologies, rather than relying solely on hard coded interfaces or ad hoc conventions. In this sense, the Semantic Web can be seen as one of the key engines driving contemporary efforts to combine symbolic knowledge representation with data-driven AI and distributed coordination. It supplies the formal scaffolding for meaning, policy, and provenance on top of which learning-based and agent-based techniques can operate in a more transparent, verifiable, and composable way [22, 21].

2.4 Stigmergy

In stigmergic coordination, individuals do not communicate peer-to-peer. Instead, they communicate through the environment by leaving and detecting signals. Whenever an individual carries out an action, it modifies the environment in some way (for example, depositing a substance, altering a physical structure, or updating a shared data repository). These modifications serve as cues that influence the subsequent behaviour of other individuals. Crucially, the communicating individuals need not be present at the same time or place, the persisting trace in the environment bridges that gap. The environment effectively functions as a shared memory or bulletin board where information is posted and read by many participants. This allows a form of asynchronous, distributed collaboration, where each member of the system responds to the state of the environment, and through these local responses, a coherent global pattern of activity emerges. There is no need for any central controller or explicit messaging between each pair of agents, coordination results implicitly from each one following the cues available in the environment [15, 48].

While ants and termites are classic natural examples, stigmergy is a domain-agnostic

principle. Any system in which agents leave state changes that influence others can exhibit stigmergic coordination. For instance, bacteria can coordinate via chemical secretions in their environment to form complex colonies or biofilms [49]. Even humans make use of stigmergy in certain contexts. Consider collaborative platforms like Wikipedia [50], where users indirectly coordinate their edits, each edit or contribution leaves an information trace that others later read and react to, gradually producing a well-organised article without a centralised plan. In all these cases, the hallmark of stigmergy is that coordination arises from the trail of work itself. The pattern of activity is guided by the accumulated modifications in the shared medium, enabling individuals to build on each other's actions even if they are not concurrently interacting.

2.4.1 Key Characteristics and Benefits

Stigmergy provides several important characteristics and advantages as a coordination mechanism [15, 51]:

- **Indirect Communication via Environment -** Instead of direct messages between individuals, information is exchanged by altering the environment. This decouples the sender and receiver of information in space and time. An individual simply responds to whatever traces are present in the environment, without needing to know who left them or to coordinate a meeting. This greatly simplifies communication in large groups, as participants do not require complex communication networks, the environment is the medium that everyone observes and influences.

- **Shared Persistent Memory -** The traces left in stigmergic systems often persist for some time, effectively creating a shared memory that outlives the initial action. For example, an ant's pheromone trail remains on the ground for a while, guiding other ants even after the original ant has gone. This persistence means the system has a form of memory of past actions, which can inform future actions. In contrast, direct signals (like one-to-one messages) are fleeting. The lingering environmental markers ensure that useful information (e.g. the path to food or progress on a task) remains available to the whole group as a point of reference.

- **Self-Organisation and Emergent Order -** Stigmergy naturally gives rise to self-organising behaviour, requiring neither a central coordinator nor a global plan. Instead, complex and organised structures emerge from the accumulation of many simple local interactions between agents and their shared environment. Each agent follows basic rules (e.g. follow stronger pheromone scent, add pheromone when you find food), and yet the aggregate outcome can be highly structured and adaptive. This emergent order is both robust and flexible. When conditions change, such as a food source becoming exhausted or an obstacle appearing, the collective behaviour adapts naturally as new traces are laid and old ones fade, without any agent explicitly re-planning the system as a whole. In essence, stigmergy allows global coherence to arise from local responses, yielding intelligent group behaviour from the bottom up.

- **Scalability and Robustness -** Because communication is indirect and decentralised, stigmergic systems tend to scale well as the number of participants grows. There is not a combinatorial explosion of message exchanges, each agent only interacts with the environment, no matter how many others are in the system. This reduces overhead and avoids the need for each agent to manage many connections.

The mechanism is robust to the loss or addition of individuals, if one agent is removed, the remaining traces in the environment still guide the others. Similarly, new agents can join and immediately pick up the cues from the environment. The system can also adapt dynamically, for example, pheromone trails evaporate over time, which allows outdated paths to weaken and new, more fruitful paths to emerge. Such features make stigmergy-driven coordination highly adaptive and scalable in complex, changing environments.

2.5 Cloud-Native Orchestration

In cloud-native architectures, distributed services are often decoupled through message-oriented microservices. Instead of direct calls, independent microservices communicate asynchronously via message brokers (e.g. RabbitMQ, Apache Kafka), which queue and forward messages. This messaging model decouples producers and consumers, allowing each service to operate at its own pace without blocking. Asynchronous queues thereby improve fault tolerance and scalability, where producers can send messages and continue operating, while consumers retrieve and process messages when ready. This loose coupling enables services to be developed, deployed, and scaled independently, making the overall system more resilient to changes or partial failures [52, 53, 54].

Another fundamental concept is DAG-based execution for orchestrating multi-step workflows. A directed acyclic graph (DAG) represents a workflow as a set of tasks (nodes) with directed edges defining their execution order and dependency constraints. By construction, a DAG has no cycles, so each task executes only once and only after its prerequisites have completed. This structure enforces a clear, one way progression of steps, preventing circular waits and ensuring correct sequencing. It also enables parallelism because independent tasks (with no mutual dependencies) can run concurrently, improving efficiency. Modern orchestration frameworks such as Apache Airflow adopt DAG based scheduling, where users define workflows as DAGs of tasks [55], and cloud services like AWS Step Functions [56] use state machine models that similarly enforce acyclic execution flows. The DAG approach thus helps manage complex workflows by making dependencies explicit and execution order deterministic.

To support running many workflows reliably, orchestration systems provide per-run (per workflow) isolation of resources. Each workflow execution is logically partitioned, with its own message queues, logs, and state, so that concurrent runs do not interfere with one another [56, 57]. In practice, an "isolated" workflow context consists of dedicated channels (e.g. separate queue names or topics) and separate logs or history for that particular run. This isolation ensures safe parallel execution, where one workflow's messages or intermediate data never get mixed into another's, and errors or crashes in one run remain contained. By partitioning runtime state per-run (often using unique workflow IDs or namespaces), the orchestrator prevents race conditions between workflows and provides fault isolation, improving the robustness of multi-tenant or multi-instance systems.

Achieving fault tolerance further requires idempotence and deduplication in distributed workflows. Idempotence means that performing the same operation multiple times has the same effect as doing it once. In other words, an idempotent service or task produces a stable, correct outcome even if it receives a duplicate message or retries an action multiple times. This property is crucial because in distributed systems, duplicate events or repeated requests are common (for example, due to network retries or at-least-once delivery guarantees) [58, 59]. Deduplication mechanisms complement idempotence by

detecting and eliminating duplicates so that each unique message or job is processed only once. Together these techniques prevent unintended side effects such as double-processing or inconsistent state updates. In practice, systems implement idempotence by various strategies. For instance, a client may attach a unique idempotency key to each request, the service records keys it has seen and ignores any subsequent request with the same key, ensuring no duplicate action occurs [60]. Similarly, database operations can be made idempotent by using "upsert" (update or insert) logic or unique constraints, so if the same operation is executed twice, the second execution simply updates the existing record and does not create a duplicate. Consumers of message queues often keep track of processed message IDs in a log, if a message with an already seen ID arrives again, it can be safely skipped. By designing every step to tolerate repeats in this way, orchestrated workflows can safely retry tasks after failures and handle at-least-once delivery without corrupting the overall process.

Finally, cloud orchestration must maintain transactional state and consistency throughout a workflow's steps. Each step often involves updating shared state or databases, and it is vital that these state transitions occur reliably despite failures. Traditional ACID (Atomicity, Consistency, Isolation, and Durability) [61, 62] transaction principles are applied so that operations are atomic (all or nothing) and consistent in effect. This means a step's set of state updates either completes entirely or, if an error occurs mid process, has no effect at all (the system rolls back to the prior state). Ensuring atomicity prevents partial updates that could leave data in an inconsistent state. Thus, if a workflow was internally consistent before a transaction, it remains consistent afterward, with no intermediate corruption. Within a single service, this is achieved using database transactions or other atomic operations (for example, a commit that moves the system from one consistent state to the next, or a compare and swap on a shared variable). In distributed multi-service workflows, maintaining consistency requires additional patterns because a single global transaction (spanning multiple services or databases) is often infeasible. Instead, cloud orchestration relies on approaches like the saga pattern [63], which breaks a cross service operation into a sequence of local transactions with compensating actions for rollbacks, or the transactional outbox pattern [64], which atomically couples database changes with message publication. These patterns ensure that updates and interservice notifications go hand in hand. For example, if a workflow step involves writing to a database and sending an event, the system will do both in one atomic unit, so either both the database update and the event emission succeed or both are aborted on failure. This guarantees that other services never observe a partial outcome (such as an event about a change that was never committed, or a database change without the corresponding notification). By using such transaction management techniques, cloud native orchestrators uphold strong consistency and reliable state transitions across the entire workflow, despite the distributed nature of the system.

From a MAS perspective, cloud-native architectures are a natural fit because they already decompose functionality into loosely coupled, independently deployable services that resemble distributed agents. Each microservice or agent encapsulates its own logic and state, communicates via messages, and can be scaled or upgraded without central coordination. Cloud-native orchestration provides the substrate for coordinating these agents at scale, where message brokers implement the communication fabric, DAG- or state-machine-based workflow engines define and enforce interaction protocols, and per-run isolation together with transactional guarantees ensure that agent interactions remain robust and consistent even under failures. This tight alignment between MAS principles and cloud-native patterns motivates adopting cloud-native orchestration as the execution backbone for complex, agent-based workflows.

2.6 Assignment Policies

Assignment policies in Multi-Agent Systems define how tasks are mapped to agents under constraints of time, capability, communication cost, and uncertainty. They can be viewed along a spectrum from simple deterministic rules to adaptive, market and score based mechanisms. This chapter outlines three representative, inclusive classes of policies that are frequently used as conceptual and algorithmic baselines [65, 66, 67].

2.6.1 First-Come, First-Served (FCFS) Assignment

First-Come, First-Served (FCFS) [68] is a basic assignment policy and a common baseline in Multi-Agent coordination. Tasks are placed in order of arrival, and when an agent becomes available, it receives the earliest pending eligible task. The three advantages of FCFS are that it is simple to implement, it has low computational cost, and it is perceived as "time-fair" because earlier requests are not passed over without reason.

FCFS can work adequately in homogeneous or lightly loaded settings, where agents have similar capabilities and tasks have comparable complexity [68]. However, once heterogeneity, deadlines, or priority differences are present, its weaknesses become clear. FCFS ignores agent capabilities, current workload, reliability, and spatial or operational context. As a result, an early, poorly matched, or unusually difficult task can tie up valuable agents, while later tasks that are more urgent or better aligned are delayed. This dynamic reduces resource utilization, increases completion times, and creates a gap between local assignment decisions and overall system objectives.

2.6.2 Improved Contract Net Protocol

The Contract Net Protocol (CNP), as discussed earlier, introduces a structured, market-like assignment process. However, classical CNP can incur significant communication overhead and redundant negotiation in large or dynamic systems due to broad task announcements, repeated bidding, and limited use of historical trust or performance information, which may also result in selecting agents that are eligible but not globally efficient or reliable. Improved Contract Net Protocols (ICNP) [5] refine this framework using ant colony inspired response thresholds and pheromone models to achieve selective, adaptive participation and more efficient coordination. Tasks are encoded as task pheromones with attributes such as priority, stimulus, and reward, while agents emit bidding pheromones that include capability, estimated completion time, current load, and a dynamic response threshold shaped by past successes, failures, and trust. An agent bids only when the task stimulus exceeds its threshold, filtering out overloaded or unsuitable candidates and reducing unnecessary communication. As agents complete tasks successfully, their thresholds adjust to reinforce appropriate task types. Repeated failures increase thresholds and suppress unsuitable bids. Coupled with mechanisms for monitoring execution and reassigning tasks when contractors overrun or fail, ICNP style schemes reduce traffic and run time, improve task completion rates, and promote more rational, self organizing, and robust task allocation while preserving the distributed negotiation structure of the original protocol [5].

2.6.3 Score-Based

Score-based assignment policies allocate tasks by computing an explicit score or utility for each possible agent–task pairing. Each pairing is evaluated on criteria such as capability

match, expected completion time, execution and communication cost, current and projected load, historical reliability or trust, task priority and deadlines, and relevant risk, energy, or other resource constraints. A centralised or distributed mechanism then chooses allocations that maximise total utility or equivalently minimise a global cost, subject to resource limits, precedence constraints, and service level objectives [65, 66].

Classical bidding schemes can be viewed as a special case that uses simple scores (for example, selecting the lowest cost bid), whereas more advanced approaches use weighted combinations of features or learned estimators to represent system objectives more accurately. These policies also support fairness and transparency, where fairness terms can be added to reduce starvation or persistent overload of particular agents, and the factors driving each decision can be reconstructed from the logged utility components. In the Multi-Agent literature, utility driven methods therefore constitute a flexible and principled family of policies that make the trade-offs in agent task matching explicit and tunable [66, 67].

2.7 Summary

This chapter has introduced the conceptual and technical foundations that underpin the proposed framework. It began with classical Multi-Agent Systems and their coordination mechanisms, where the Contract Net Protocol illustrated how decentralised task allocation can be achieved through structured negotiation. The discussion then moved to LLM-based Multi-Agent Systems, where natural language driven agents enriched with tools, memory, and human-in-the-loop oversight extend the classical paradigm with new flexibility but also introduce new fragilities. The Semantic Web stack was presented as a machine-interpretable substrate for representing knowledge, constraints, and provenance, providing the formal grounding required for auditable coordination. Stigmergy was introduced as a principle of indirect, environment-mediated coordination that enables adaptive and self-organising collective behaviour without central control. Cloud-native orchestration patterns were then summarised as the execution substrate that makes Multi-Agent coordination practically realisable at scale, and the chapter closed with a review of representative task-assignment policies that anchor the design space in which the proposed framework is positioned.

Together, these foundations supply the vocabulary and the conceptual tools needed to characterise the limitations of current orchestration platforms and to motivate the architectural choices that follow. In the following chapter, the state of the art in LLM-based orchestration is examined in depth, focusing on LangGraph and CrewAI as representative case studies, in order to map their architectural limitations against the three gaps that the proposed framework addresses.

3. State of the Art

As large language models have improved at reasoning, developers have started building frameworks that let multiple AI agents work together to tackle complex, multi-step problems. Early projects such as AutoGPT [41] and BabyAGI [69] demonstrated the potential of chaining autonomous agents across extended task horizons, but these systems were primarily experimental and exposed significant challenges in reliability, controllability, and evaluation, as surveys of LLM-based Multi-Agent Systems confirm [17]. In response, more structured orchestration frameworks have emerged in both industry and research, including LangGraph [18], CrewAI [19], and AutoGen [20], each supporting Multi-Agent workflows through different architectural philosophies. More specialised platforms such as ChatDev [70], MetaGPT [71], and HuggingGPT [40] explore domain-specific forms of agent collaboration, but remain oriented toward experimentation rather than standardised production deployment.

This chapter provides an in-depth analysis of LangGraph and CrewAI, which represent two mature, complementary points in the current design space. LangGraph exemplifies graph and state-centric orchestration with strong guarantees around control, observability, and persistence, while CrewAI embodies a role-based, crew-oriented approach that emphasises developer ergonomics and collaborative agent workflows. Both are actively maintained, well documented, and used in non-trivial applications, making them representative case studies for analysing orchestration trade-offs. Other systems such as AutoGen, ChatDev, MetaGPT, and HuggingGPT are therefore treated as important reference points rather than primary objects of study, as they either target narrower domains, remain more exploratory in nature, or currently offer less comprehensive operational and governance capabilities.

3.1 LangGraph

This section details LangGraph using three main viewpoints: Architecture, Coordination, and Runtime. Architecture covers the main components and overall design of the framework. Coordination describes how tasks are divided among agents and how their work is synchronised or communicated. Runtime refers to how execution is scheduled, for instance, whether the system runs agents concurrently or in sequence, and how it manages long-running processes or failures.

3.1.1 Architecture

LangGraph uses a graph-based architecture where workflows are represented as directed graphs of nodes and edges. Each node is a single step in the agent's workflow, for example, calling an LLM, using a tool, or running custom code. Edges between nodes define how control moves from one step to the next, and these transitions can be fixed or conditional on the current state. This means the workflow is not constrained to a single fixed sequence, as its path can instead be determined dynamically at runtime [72].

LangGraph keeps a central shared state that all nodes can access. Because the graphs are stateful, each node can read from and write to this shared state, letting the agent remember intermediate results and context across the whole workflow. Workflows can also include cycles (loops), not just one-way flows, which allows for iterative behaviour when needed.

Overall, the architecture is built from three primitives, nodes as actions, edges as transitions, and shared state, giving developers fine-grained control over how an agent's process is structured.

3.1.2 Coordination

By modelling Multi-Agent or multi-step workflows as a single graph with shared state, LangGraph makes it easier to tightly coordinate specialised agents and sub-tasks. Its graph model naturally supports parallel work, branching, and synchronisation.

Because the same node can have multiple successors, a workflow can fan out into parallel branches. For example, if two sub-tasks are independent, the graph can have two outgoing edges from one node so that both tasks run at the same time. The runtime handles synchronisation by ensuring that any node depending on several previous nodes waits until all of them have finished before it begins execution [73, 74]. This enforces explicit join semantics and reduces coordination hazards, ensuring that parallel branches synchronise in a controlled manner.

LangGraph also supports dynamic decision-making through conditional edges. An edge can look at the current state and choose the next node based on some condition (for example, taking one path if an intermediate result meets a threshold, and another path otherwise). This allows the agent to break down or redirect tasks on the fly.

In addition, the graph can express loops by connecting an edge back to an earlier node, enabling iterative refinement. In more advanced setups, developers can implement patterns like an orchestrator–worker model within the graph, for instance, having one node act as an orchestrator that spawns several worker nodes to handle sub-tasks in parallel, using LangGraph's APIs for dynamic node creation.

3.1.3 Runtime

LangGraph uses a stepwise execution model inspired by graph processing frameworks like Pregel [73, 74] and dataflow systems. When an agent's graph is invoked, the runtime starts at a special START node and determines which nodes are ready to run based on the current state and edge conditions. In each iteration (often called a "superstep"), all eligible nodes (for example, the START node or any whose predecessors have finished) are executed, possibly in parallel if there are multiple active nodes. The scheduler can therefore run several node functions at the same time, then wait for all of them to complete.

After each batch of executions, LangGraph updates the shared global state with the outputs from those nodes and creates a checkpoint of the entire state [75]. This checkpointing is important because it provides durability and fault tolerance, allowing the workflow to resume from the latest state after a failure instead of starting over. It also supports pausing an agent mid-execution (for example, to wait for human input) and resuming later, since the exact workflow state is preserved.

The scheduler then plans the next step by checking which edges were activated by the recent outputs, which determines the next set of nodes to run. This plan $\rightarrow$ execute $\rightarrow$ update cycle repeats until no further nodes are triggered. In this way, LangGraph's runtime enforces a controlled and deterministic progression through the graph, where nodes execute only when their preconditions are satisfied as defined by the graph structure and the current state.

Developers also have fine-grained control over scheduling. They can insert human-in-the-loop steps, add custom synchronisation points, or tune persistence settings, making

LangGraph suitable for long-running or interactive agent sessions. The trade-off is that using the runtime effectively requires some understanding of concurrency and state management, since the execution flow is a coordinated graph traversal rather than a simple linear script.

3.2 CrewAI

CrewAI is analysed next through the same three viewpoints of architecture, coordination, and runtime.

3.2.1 Architecture

CrewAI uses a role-based Multi-Agent architecture, organising an AI system as a "crew" of specialised agents. Each agent is assigned a specific role, a persona with defined skills and responsibilities, and operates semi-autonomously. For example, a Researcher agent focuses on information gathering, while a Writer agent synthesises findings into a report. Agents can also be equipped with role-appropriate tools or APIs (e.g. a database search tool for the Researcher). This combination of specialisation and tool access aligns with ReAct's task-specific grounding and supports more focused, interpretable, and reliable behaviour. More broadly, findings on role-based prompting and expert-persona framing suggest that LLM performance improves when the assigned roles are well aligned with the task's demands [35].

In CrewAI, tasks are first-class objects. A Task defines a unit of work with a description, an expected output, and an assigned agent that will carry it out. A group of agents and their tasks are organised into a Crew, which represents the overall collaborative workflow targeting a higher-level goal. The crew is managed by a Process, which acts as the orchestrator or project manager for the group. The Process decides how tasks are scheduled, assigned, and executed according to a chosen strategy.

These components form a simple hierarchy in which the Crew contains agents working on tasks, while the Process coordinates and monitors their collective activity.

Each agent also maintains its own memory and context tuned to its role. CrewAI supports several memory layers, including short-term memory for a single task or conversation, long-term memory that persists across sessions, and entity-specific memory for tracking information about particular people or concepts. This allows, for instance, the Researcher agent to accumulate facts over time, while the Writer agent builds up knowledge about style and previous content, both working toward the same overall objective [19].

3.2.2 Coordination

In CrewAI, coordination is centred on dividing work by roles and managing hand-offs between agents. Instead of a single chain of actions, CrewAI encourages a division of labour, where each agent handles the part of the problem that matches its role. Tasks are typically assigned to the agent best suited for that subtask, as in the common pattern where a Researcher agent works first and a Writer agent follows. The framework ensures that these tasks run in the right order and stay properly synchronised.

A Process (either an implicit controller or an explicit manager agent) is responsible for sequencing tasks and enforcing dependencies. For example, if the Writer's task depends on the Researcher's findings, the Process will not start the writing task until the research task is complete and its results are available.

CrewAI natively supports both sequential and parallel workflows. You can define tasks that run one after another, or launch multiple tasks at the same time if they are independent. To scale up, developers can even create multiple agents with the same role to work in parallel on different parts of a problem. The framework then synchronises their outputs and aggregates results as needed.

Agents in a crew can also communicate and delegate among themselves, which is central to CrewAI's coordination model. They can ask each other questions or pass intermediate results through shared context, much like coworkers consulting one another. For instance, an agent might call a tool or function exposed by another agent to tap into its expertise. CrewAI supports this by maintaining a global context (or "contextual memory") shared across the crew, so that information discovered by one agent can inform the others.

CrewAI also makes it easy to include human oversight in the loop. Tasks can be configured to require human approval or input before continuing (for example, reviewing the Researcher's notes before the Writer proceeds). The system pauses at these checkpoints, lets a human operator intervene, then resumes the workflow, helping ensure that agents stay aligned with both their roles and the user's intent [19, 76].

3.2.3 Runtime

CrewAI's runtime executes agent tasks using a managed, hierarchical scheduling model. When a crew is launched, the framework may internally create a manager agent (or use a defined Process) that controls the overall execution. The manager plans the overall strategy, deciding which task to initiate first, when to trigger follow-up tasks, and how to distribute work among agents. In a simple sequential setup, the manager runs the first task, waits for it to finish, and then passes its output to the next agent. For tasks that can run in parallel, CrewAI can schedule multiple tasks concurrently when tasks are independent [76]. The runtime tracks each task's state (pending, in progress, done) and collects their results when they complete.

CrewAI introduces Flows to give developers finer control over scheduling and triggers. A Flow is an event-driven pipeline that can run alongside a crew or on its own. It lets you define explicit execution paths with conditions, loops, and branches, similar to a traditional workflow engine. For example, a Flow attached to a Crew could express logic such that if a particular data condition is met, two agents are notified in parallel and the system waits for both to complete, whereas otherwise execution loops back to another agent for a further attempt. Under the hood, Flows manage state transitions and event triggers so that the right agent is invoked at the right time, adding a layer of deterministic control on top of autonomous agent behaviour [76].

At runtime, CrewAI supports asynchronous operation by allowing crews to run non-blockingly, and multiple crew instances can execute concurrently, enabling parallel operation patterns [19, 76]. Throughout execution, the scheduler ensures agents stay within their roles, while the manager or Process intercepts outputs and routes them to the appropriate next step. When an agent finishes its task, the scheduler either advances to the next task or waits for all parallel tasks to complete before triggering aggregation or subsequent steps. This hierarchical control, an overseeing manager coordinating autonomous agents, is designed to keep the team aligned with the overall goal. While CrewAI's default scheduling is more implicit than LangGraph's graph-based runtime, the introduction of Flows allows similarly fine-grained control when needed [19, 76].

3.3 Comparisons

Both LangGraph and CrewAI enable complex multi-LLM applications, but they differ significantly in philosophy and ideal use cases.

3.3.1 Control and Abstraction

LangGraph offers explicit control over execution because workflows are defined as graphs of nodes and transitions over shared state. Developers explicitly define every node and transition, yielding full visibility into each agent step and fine-grained control over decisions. This makes it possible to intervene or debug at any point in the workflow, which is critical for long-running or safety-critical deployments. CrewAI, by contrast, provides a higher-level abstraction, where one describes agent roles and overall process, and the framework handles the low-level sequencing. This leads to faster development cycles and more concise definitions, at the cost of some direct control over intermediate steps. In CrewAI, much of the control over what happens next is managed implicitly by the agents' interactions or the preset process logic, whereas in LangGraph nothing is implicit, every transition must be encoded. The introduction of Flow constructs in CrewAI has narrowed this gap by allowing developers to insert explicit routing logic within a crew, but generally LangGraph is well suited when maximum determinism and customisability of the workflow are required [72, 19].

3.3.2 Information Flow and State Management

Another contrast is how each handles the flow of information between steps. LangGraph uses an explicit persistent state object that accumulates and carries data through the graph. This makes complex context management straightforward, as the state can be checkpointed, inspected, or modified at any point and persists even when the process pauses or retries [75]. It natively supports branching and looping, meaning an agent can dynamically choose a different path or repeat a step based on the state [72]. CrewAI relies on a shared context (often the conversation history or shared memory) to maintain continuity between agents [19]. This is sufficient for many collaborative tasks, but deep state or long-term memory is less central. If an agent needs to preserve information, it typically appends it to the shared context in natural language form. LangGraph's approach is thus better suited for workflows that require robust stateful behaviour (for example, iterative problem-solving where each iteration's outcome informs the next). CrewAI's approach is best suited to tasks that can be broken into well-defined chunks without extensive backtracking. It allows some parallelism and tends to be more linear in execution, aside from any branching logic encoded via flows [76]. LangGraph is well suited for adaptive or non-linear processes where the workflow might revisit earlier steps or handle unpredictable paths, whereas CrewAI is a strong fit for structured pipelines or role-based teams following a planned strategy [72, 19].

3.3.3 Strengths and Trade-offs

LangGraph's strength lies in its expressiveness and reliability for complex scenarios. It can model a wide range of workflow topologies and provide precise control over execution, including error handling, timeouts, and human-review points. It also naturally facilitates transparency, since one can log the entire state and path of execution for analysis. The trade-off is higher complexity and a steeper learning curve, building a LangGraph workflow

requires more engineering effort and understanding of the graph paradigm. On the other hand, CrewAI's strength is developer productivity and clarity. The concept of a crew of agents with roles is easy to grasp, and one can leverage the built-in patterns (sequential flows, manager oversight, etc.) to get a Multi-Agent System working without delving into low-level logic. This makes CrewAI well suited for rapid prototyping or use cases where the structure of collaboration is more important than fine-tuned control of each step. Its role-based design also maps well to certain real-world team structures, making it intuitive to assign, for instance, a "Planner" agent and an "Executor" agent and let them interact. The downside is that extremely complex coordination logic might be harder to implement if it doesn't fit one of the provided paradigms. For example, achieving the same level of dynamic looping or conditional branching as LangGraph might require creative use of flows or might simply be more cumbersome. In summary, CrewAI trades some power for simplicity, whereas LangGraph trades simplicity for power [72, 19].

3.3.4 Ethics and Trustworthy AI

A critical dimension where both frameworks offer limited built-in support is ethics and trustworthy AI. Although each allows developers to insert human-in-the-loop steps, neither provides standardised governance primitives such as mandatory policy gates, structured risk routing, provenance-by-default, or enforceable constraints as first-class orchestration constructs. Instead, developers must implement such safeguards manually when required. This retrofit approach can lead to inconsistent protections, where some teams may introduce comprehensive logging, bias checks, and approval workflows while others omit them entirely. In production AI systems handling sensitive data or consequential decisions, such variability introduces significant ethical and governance risks [8, 23].

Moreover, even when safeguards are added, aspects of the underlying design can increase the burden of enforcing them correctly. LangGraph's shared state model increases the burden on developers to implement strict context isolation between clients or workflows. CrewAI's reliance on shared conversational context similarly creates potential information-leakage risks if session boundaries and memory handling are not carefully controlled [23]. The limitations described above are not failures of capability but reflect orchestration primitives that prioritise flexibility and developer convenience over governance guarantees.

As regulatory frameworks such as the EU AI Act [2] introduce obligations related to risk management, transparency, and human oversight, the absence of governance-aware orchestration primitives becomes a growing concern for teams deploying Multi-Agent AI systems in regulated environments.

3.3.5 Limitations

Despite the advances offered by LangGraph and CrewAI, there are clear limitations that motivate new orchestration strategies. LangGraph, while powerful, can be onerous to implement and maintain for large graphs, and its low-level nature may overburden developers when rapid changes or experimentation are needed. CrewAI, conversely, provides an easier starting point but sacrifices some control and transparency. Developers may find it challenging to debug nuanced issues or enforce certain execution orderings because of the framework's higher-level abstraction [72, 19]. More generally, existing frameworks tend to focus either on strict structure or on maximal flexibility, but few offer both simultaneously. This opens an opportunity for a new orchestration framework that combines the precise control of a graph-based approach with the modular clarity and ease of use

of a Multi-Agent crew model. The next chapter introduces a framework built around a directed acyclic graph (DAG) paradigm that seeks to unify these strengths while addressing the shortcomings of current solutions. The goal is to enable developers to design LLM-driven workflows that are both highly controllable and readily extensible, closing the gap between LangGraph's robustness and CrewAI's user-friendly modularity. Critically, the proposed framework treats ethics as a first-class architectural concern. Fairness constraints, decision provenance, human-in-the-loop triggers, and input sanitisation are orchestration primitives, not optional add-ons. Developers using this approach obtain ethical guarantees by default, whereas with LangGraph or CrewAI they must explicitly take care of these considerations themselves [8, 2].

4. Proposed Method and Framework

This chapter presents the design of the proposed Multi-Agent coordination framework, grounded in stigmergy as a principle of indirect coordination through signals left in a shared environment [77, 15]. The chapter begins by stating the design goals that guided the framework, then describes its overall architecture and details each major component. Throughout, formal ontology-driven concepts are used, such as stig:TaskSpec for task templates, stig:Slot for required roles, and various stig:Stigma marker subtypes, to define the coordination model. The emphasis of this chapter is on what the framework proposes and why, covering its conceptual mechanisms, formal procedures, and algorithmic contributions.

4.1 Design Goals

The framework is driven by five core design goals, each addressing a limitation identified in the analysis of existing Multi-Agent orchestration platforms (Chapter 3). Together they operationalise the thesis objectives (O1–O6, Chapter 1) and serve as the evaluation criteria against which the prototype is assessed in Chapter 6:

1. **Decentralised coordination through competitive task allocation and shared environmental signals.**
   Agents must coordinate through both competitive bidding via the Improved Contract Net Protocol (ICNP) and shared environmental signals (affordance markers), rather than through direct peer-to-peer messaging or a centralised scheduler [15, 77]. ICNP ensures that task assignments reflect agent capability and availability, while affordance markers carry accumulated performance history across pipeline runs, enabling the system to reinforce successful allocation patterns over time. Together, these mechanisms ensure that the system scales gracefully with the number of agents and tasks and that no single component becomes a coordination bottleneck. This goal corresponds to objectives O1 and O2.
2. **Formal ontological knowledge substrate.**
   All core entities, including tasks, agents, capabilities, plans, and coordination markers, must be defined via a formal OWL/RDF ontology using the `stig:` namespace. This enables machine-readable specifications, automated constraint validation, and semantic interoperability across heterogeneous agent populations [9, 22]. This goal corresponds to objective O3.
3. **Safety and human oversight.**
   The framework must embed safety mechanisms as first-class architectural concerns, not optional add-ons. This includes mandatory content filtering at intake, risk-based routing to human approval workflows, ethical review gates before execution, and structured escalation paths for runtime failures [8, 2]. This goal corresponds to objective O4.
4. **End-to-end audit traceability.**
   Every decision, state transition, and agent action must be captured in a structured audit trail with end-to-end correlation identifiers. This supports post-hoc analysis, compliance verification, and explainability of system behaviour [10, 2]. This goal corresponds to objectives O3 and O4.
5. **Adaptive failure recovery and strategic replanning.**

The framework must respond to runtime failures and changing conditions through self-healing mechanisms (bounded retries, pheromone-driven re-weighting) [78, 61] and LLM-assisted strategic replanning, escalating to human operators only when automated recovery is insufficient. This goal corresponds to objective O5. The empirical evaluation of this and all other design goals is covered by objective O6.

These goals inform every architectural decision described in the remainder of this chapter and serve as evaluation criteria for the experimental evaluation in Chapter 6. Their measurable impact is tested through the targeted ablation experiments reported in Section 6.2, where each design goal is mapped to one or more experimental hypotheses.

4.2 System Architecture Overview

Figure 5 illustrates the high-level architecture of the proposed framework. The system processes a request through a pipeline of five major phases, preceded by a one-time Agent Registration step, for a total of six steps.

1. **Agent Registration** – autonomous agents register their identities and capabilities in a shared registry (Section 4.3).
2. **Task Input and Normalisation** – user-submitted tasks are validated, filtered, and transformed into structured `TaskSpec` objects (Section 4.4).
3. **Stigmergic Task Assignment (ICNP)** – agents bid on tasks through an Improved Contract Net Protocol [5] mediated by stigmergic markers (Section 4.5).
4. **Plan Generation and Constraint Validation** – bids are synthesised into an execution graph, validated against formal constraints, and transformed into an executable DAG (Section 4.6).
5. **Human Validation and Approval** – risk-based routing ensures human oversight for high-risk plans (Section 4.7).
6. **Execution, Adaptation and Failure Handling** – the plan is executed under the supervision of an LLM-based strategist and a pheromone engine that enable adaptive coordination, with structured escalation for persistent failures (Sections 4.8–4.9).

A cross-cutting **Audit & Traceability** layer (Section 4.10) captures structured records from every phase, providing end-to-end observability.

4.3 Agent Registration and Capability Model

Before any task can be processed, participating agents must register their presence and capabilities with the system (Step (a) in Figure 5). Agents and their capabilities are modelled using the ontology as follows:

- Each agent is identified by a unique URI [79] and is an instance of `stig:Agent`.
- Each agent declares a set of `stig:Capability` instances, representing discrete skills or services it can perform (e.g. `stig:ImageOCR`, `stig:RoutePlanning`).
- Each agent carries stigmergy tags (quantitative metadata modelled as `stig:Stigma` markers) that encode domain expertise, historical performance, and operational state. The tag values are intensities analogous to pheromone concentrations [77].

The Agent Registry acts as the initial stigmergic medium [15], serving as the shared environment that task-assignment processes later query. To maintain currency, the framework applies a *decay* function to agent tags over time. If an agent has not been active for a period Δt, its tag intensities are reduced by a decay factor λ:

$$I(t + \Delta t) = I(t) \cdot e^{-\lambda \Delta t} \tag{1}$$

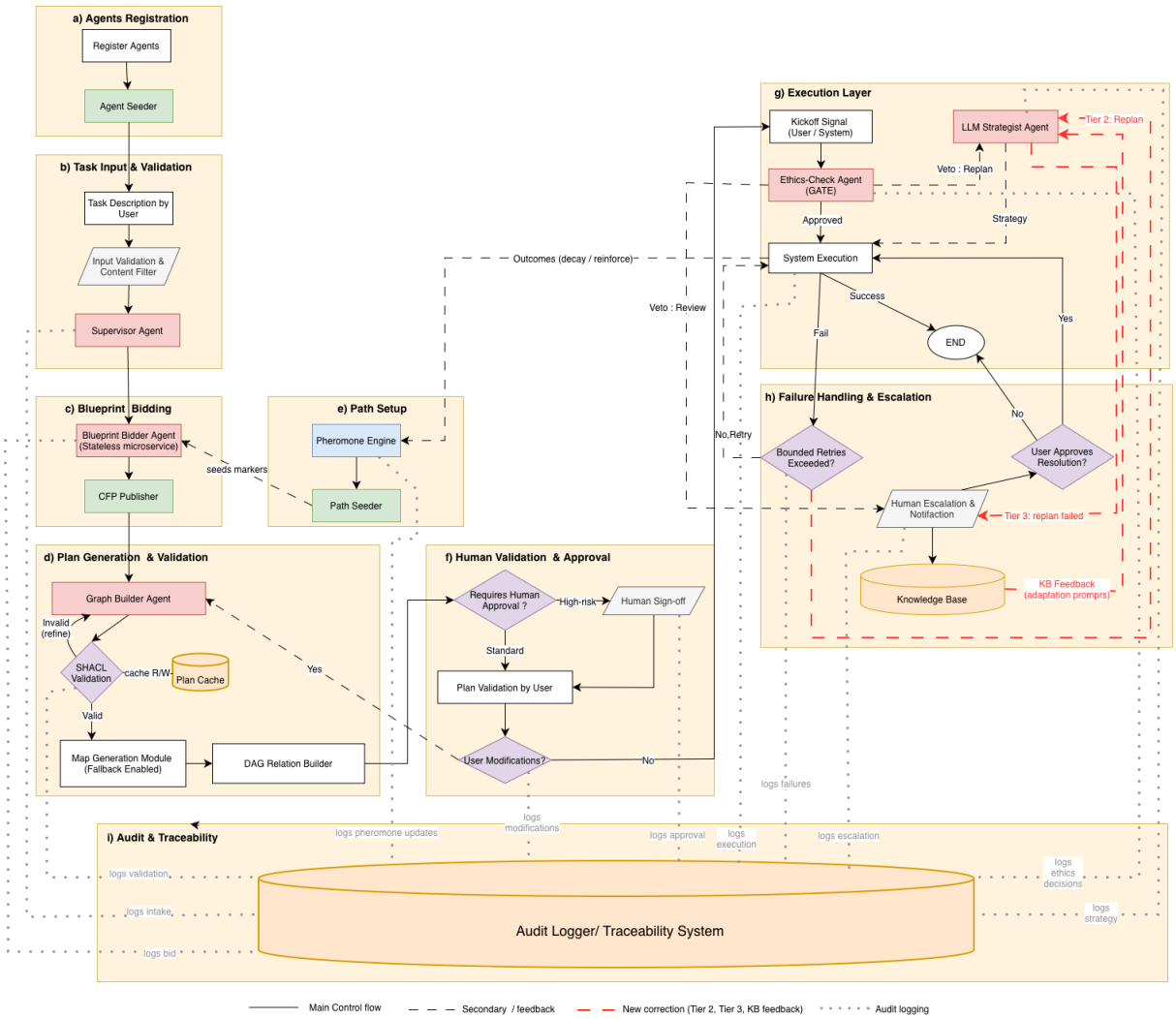


Figure 5: High-level architecture of the proposed Multi-Agent coordination framework. Phases (a)–(g) are described in Sections 4.3–4.10.

where $I(t)$ is the tag intensity at time t and $\lambda > 0$ is the decay rate. This mechanism ensures that stale agent information fades, analogous to pheromone evaporation in biological systems [77, 80], and that the registry reflects the current operational landscape at any given time. When a task specification declares a required capability for a `stig:Slot` (via `stig:requiresCapability`), the system queries the registry to identify candidate agents whose capabilities match the requirement and whose tag intensities exceed a minimum threshold.

4.4 Task Normalisation: From Unstructured Input to TaskSpec

The Task Input and Normalisation subsystem (Step (b) in Figure 5) transforms heterogeneous user-submitted task descriptions into well-formed `stig:TaskSpec` objects suitable for downstream processing. A three-stage normalisation procedure is defined (Algorithm 1).

Algorithm 1 Task Normalisation Procedure

Require: User input payload P
Ensure: Validated `TaskSpec` object T or rejection

Stage 1: Reception and Metadata Extraction
$M \leftarrow \textsc{ExtractMetadata}(P)$
if M is missing required fields **then**
  return $\textsc{Reject}(\text{“Missing required metadata”})$
end if
Stage 2: Content Filtering
$filterResult \leftarrow \textsc{ContentFilter}(P)$
if $filterResult = \textsc{Blocked}$ **then**
  return $\textsc{Reject}(\text{“Content policy violation”})$
end if
Stage 3: Structuring
if P contains a well-formed `TaskSpec` **then**
  $T \leftarrow \textsc{ParseTaskSpec}(P)$
else if LLM provider is available **then**
  $prompt \leftarrow \textsc{BuildSchemaPrompt}(P, M)$
  $response \leftarrow \text{LLM.}\textsc{Generate}(prompt)$
  $T \leftarrow \textsc{ParseJSON}(response)$
  if T is not valid **then**
    $T \leftarrow \textsc{HeuristicExtract}(P)$ ▷ Fallback
  end if
else
  $T \leftarrow \textsc{HeuristicExtract}(P)$
end if
$\textsc{Persist}(T, \text{graphStore})$
return T

4.4.1 Stage 1: Reception and Metadata Extraction

The system accepts task descriptions in multiple formats, including natural language text, structured JSON, or hybrid payloads. At this stage, the system extracts and validates required contextual metadata (pipeline identifiers, user context). The absence of required metadata results in immediate rejection with a structured error response.

4.4.2 Stage 2: Content Filtering and Safety Gate

Before any further processing, the input passes through a content filter that enforces organisational policies and ethical guidelines. Tasks that fail content filtering (e.g. containing harmful, prohibited, or policy-violating content) are rejected immediately [23]. This gate ensures that the coordination pipeline never processes inappropriate requests, serving as the first safety checkpoint in accordance with Design Goal 3 (Section 4.1).

4.4.3 Stage 3: LLM-Assisted Structuring

If the input does not already conform to the `TaskSpec` schema, a Supervisor Agent invokes a Large Language Model (LLM) to extract and structure task information from the natural language description. The LLM is prompted to produce a JSON object conforming to the `TaskSpec` schema, which includes the fields shown in Table 1.

If the LLM fails to produce valid output, or if no LLM provider is configured, the system falls back to a rule-based heuristic approach that extracts available information from the payload fields. The resulting `TaskSpec` objects are persisted to the graph marker store for downstream consumption.

Field	Type	Description
`task_id`	URI	Unique task identifier
`description`	String	Human-readable task description
`required_capabilities`	List⟨`stig:Capability`⟩	Capabilities needed to fulfill the task
`tags`	List⟨String⟩	Domain and classification tags
`risk_level`	Enum{low, medium, high}	Risk classification
`priority`	Enum{low, medium, high, critical}	Risk classification
`requires_human_approval`	Boolean	Whether human sign-off is mandatory

Table 1: Core fields of a `stig:TaskSpec` object.

4.5 Stigmergic Task Assignment via Improved Contract Net Protocol

The task assignment phase (Step (c) in Figure 5) matches validated `TaskSpec` objects with registered agents through a decentralised bidding mechanism. The Improved Contract Net Protocol (ICNP) introduced by Zhang et al. [5] is adopted as the conceptual basis for task assignment, integrated with the stigmergic marker infrastructure to create a coordination mechanism that is both autonomous and observable. The framework retains Zhang's core mechanism, namely a stimulus-threshold admission model with dynamic threshold updates driven by success history. The closed-form expressions used in this work are reformulated. The admission gate (Eq. 2) folds agent load into the stimulus score rather than the bid score. The threshold update (Eq. 3) replaces Zhang's task-difficulty terms with a single learning-rate adjustment against the system-wide success mean. The bid score (Eq. 5) replaces Zhang's history factor Y with an explicit capability-match and trust decomposition. Two further additions, the exponential decay of Eq. 1 and the clamped reinforcement of Eq. 6, integrate Zhang's mechanism with the broader ACO-derived stigmergic substrate of Dorigo and Stützle [80].

4.5.1 Rationale for ICNP

Section 2.6 surveyed three representative assignment policy classes, namely FCFS, score-based, and ICNP, each occupying a distinct position in the autonomy-optimality trade-off space. The choice among them is governed by the framework's design goals as outlined in Section 4.1.

- FCFS is incompatible with Design Goal 1 (Decentralised Coordination) because, while nominally distributed, it ignores agent capabilities and environmental signals, producing assignments that degrade under heterogeneous workloads as discussed in Subsection 2.6.1.

- Score-based allocation (Subsection 2.6.3) can yield globally optimal assignments, but its reliance on a central coordinator conflicts with Design Goal 1 and introduces a single point of failure that undermines Design Goal 5 (Adaptive Resilience).
- ICNP (Subsection 2.6.2), by contrast, is decentralised by construction, as agents bid autonomously based on local evaluation of task stimuli, the system selects winners through multi-criteria ranking, and pheromone-driven threshold adaptation provides the self-organising dynamics required by Design Goal 5.

Beyond these alignment properties, ICNP offers two additional advantages for the proposed framework. First, each bidding round produces explicit, machine-readable bid records that feed directly into the Audit and Traceability layer (Design Goal 4). Second, the protocol's affordance markers integrate naturally with the stigmergic coordination medium described in Section 4.3, allowing bidding outcomes to persist as environmental signals that inform downstream plan generation. These properties make ICNP the appropriate coordination mechanism for the framework, as it satisfies all five design goals without requiring a centralised authority.

4.5.2 Call for Proposals (CFP)

For each validated `TaskSpec` τ, the system issues a Call for Proposals containing:

- Required capabilities: $\tau.\mathit{required\_capabilities}$
- Priority level: $\tau.\mathit{priority}$
- Stimulus intensity: s_0 (initial attractiveness of the task)
- Reward value: r (value proposition for participating agents)
- Time-to-live: TTL (validity window for bids)

If no agents respond within the TTL, the system initiates a new round with increased stimulus intensity: $s_{k+1} = s_k \cdot (1 + \alpha)$, where $\alpha > 0$ is the stimulus amplification factor. This ensures that tasks that initially fail to attract bids become progressively more attractive.

4.5.3 Agent Evaluation and Response Threshold

Each registered agent a_i evaluates open CFPs by computing a *stimulus score* that captures how well positioned the agent is to accept additional work:

$$S(a_i, \tau) = w_{\text{urg}} \cdot u(\tau) + w_{\text{avail}} \cdot \left(1 - \frac{\mathit{load}(a_i)}{\mathit{maxLoad}(a_i)}\right) \tag{2}$$

where $u(\tau) \in [0, 1]$ is the task's urgency derived from its stimulus intensity and priority, and $\mathit{load}(a_i)/\mathit{maxLoad}(a_i) \in [0, 1]$ is the agent's current occupancy fraction. The weights satisfy $w_{\text{urg}} + w_{\text{avail}} = 1$. The stimulus score answers whether the agent can take on this task, whereas the bid score B defined in Eq. 5 of Subsection 4.5.4 answers whether the agent is the best fit among those who can. Keeping the two separate prevents fit signals from dominating the admission decision and conversely prevents load from distorting fit comparisons among admitted bidders.

Each agent maintains an individual *response threshold* θ_i that determines whether it submits a bid at all, as a bid is issued only when $S(a_i, \tau) > \theta_i$. The threshold is dynamic.

$$\theta_i^{(t+1)} = \theta_i^{(t)} + \beta \cdot (\mathit{successRate}_i - \bar{s}) \tag{3}$$

where β is a learning rate and $\bar{s}$ is the system-wide average success rate. Agents with above-average success raise their thresholds and become more selective, while underperforming agents lower theirs and become more willing to bid, producing a self-regulating population dynamic [51].

4.5.4 Bid Generation and Selection

When the admission condition $S(a_i, \tau) > \theta_i$ is satisfied, the agent generates a bid whose composite *bid score* combines three fit primitives that characterise the agent-task pair. The first is the *capability match* $capMatch(a_i, \tau) \in [0, 1]$, given by the fraction of required capabilities the agent possesses.

$$capMatch(a_i, \tau) = \frac{|a_i.caps \cap \tau.requiredCaps|}{|\tau.requiredCaps|} \tag{4}$$

where $a_i.caps$ denotes the set of capabilities possessed by agent a_i and $\tau.requiredCaps$ the set of capabilities required by task τ. The second primitive is *trust* $trust(a_i) \in [0, 1]$, which reflects the agent's historical task success rate over a sliding window and is reinforced through the marker feedback loop discussed later in this section. The third primitive is $cost(a_i, \tau)$, the agent's estimated resource expenditure for task τ. Together, these primitives form the inputs to the bid score.

$$B(a_i, \tau) = \gamma_1 \cdot capMatch(a_i, \tau) + \gamma_2 \cdot trust(a_i) + \gamma_3 \cdot \left(1 - \frac{cost(a_i, \tau)}{maxCost}\right) \tag{5}$$

where $\gamma_1, \gamma_2, \gamma_3$ are scoring weights with $\gamma_1 + \gamma_2 + \gamma_3 = 1$. Load does not appear here because it has already been enforced upstream by the admission gate in Eq. 2, so B is a pure fit score over capability, trust, and cost-effectiveness. The prototype described in Subsection 5.4.3 extends Eq. 5 with additional runtime signals, including a historical success-rate bonus, a speed-against-baseline bonus, a failure-surface penalty, and a learned bandit-exploration term, which sharpen ranking without changing the underlying decision structure. Pheromone intensity enters the bid score indirectly through threshold adaptation in Eq. 3 and through the trust signal fed back from marker reinforcement.

The system collects all bids for each task, ranks them by B, and selects the top-ranked bid (or multiple bids when redundancy is desired). Successful bids produce *affordance markers*, `stig:AffordanceMarker` instances in the graph store, which encode agent-task pairings together with their bid scores. These markers are the stigmergic traces [15] that guide subsequent plan generation.

Affordance markers come in two kinds. *Exploratory traces*, produced by path seeding and in-flight bidding feedback, undergo continuous exponential decay $I(t + \Delta t) = I(t) \cdot e^{-\lambda \Delta t}$, implementing the forgetting of stale preferences so the system can adapt to changing conditions. *Committed assignment markers*, created when a successful (task, agent) pairing completes execution, and are exempt from time-based decay but remain modifiable by failure feedback and by human or system-level plan adjustments. This two-tier lifecycle preserves stigmergy's adaptive properties while giving validated assignments the stability needed for downstream planning and audit.

Algorithm 2 presents the complete ICNP bidding procedure. One implementation detail is worth highlighting. The eligibility check $capMatch(a_i, \tau) > 0$ on line 9 does *not* use the strict set-intersection form of Eq. 4. Instead, it applies a semantically expanded notion of capability matching (for example, an agent declaring `python` is considered to match a task requiring `programming`), so that viable specialists are not excluded on purely lexical grounds. The strict form is reserved for computing the *capMatch* value that feeds into the bid score B on line 10, where exact matches should carry more weight than loose semantic overlaps.

Algorithm 2 Improved Contract Net Protocol (ICNP) Bidding

Require: Set of validated tasks $\mathcal{T}$, set of registered agents $\mathcal{A}$, max rounds K
Ensure: Set of affordance markers $\mathcal{M}$ (agent-task assignments)

$\mathcal{M} \leftarrow \emptyset$
for each task $\tau \in \mathcal{T}$ **do**
  $s \leftarrow s_0(\tau)$ ▷ Initial stimulus intensity
  for $k = 1$ to K **do**
    $cfp \leftarrow \textsc{IssueCFP}(\tau, s)$
    $bids \leftarrow \emptyset$
    for each agent $a_i \in \mathcal{A}$ **do**
      $S_i \leftarrow \textsc{ComputeStimulus}(a_i, \tau)$ ▷ Eq. 2
      if $S_i > \theta_i$ **and** $capMatch(a_i, \tau) > 0$ **then**
        $B_i \leftarrow \textsc{ComputeBidScore}(a_i, \tau)$ ▷ Eq. 5
        $bids \leftarrow bids \cup \{(a_i, B_i)\}$
      end if
    end for
    if $bids \neq \emptyset$ **then**
      $ranked \leftarrow \textsc{SortDescending}(bids)$
      $a^* \leftarrow ranked[0].agent$ ▷ Best bidder
      $m \leftarrow \textsc{CreateAffordanceMarker}(\tau, a^*, ranked[0].B)$
      $\mathcal{M} \leftarrow \mathcal{M} \cup \{m\}$
      break ▷ Task assigned, proceed to next task
    else
      $s \leftarrow s \cdot (1 + \alpha)$ ▷ Increase stimulus for next round
    end if
  end for
end for
return $\mathcal{M}$

4.6 Plan Generation, Constraint Validation, and DAG Construction

The Plan Generation and Validation phase (Step (d) in Figure 5) synthesizes task specifications and bidding results into a coherent execution plan, validates the plan against formal constraints, and transforms it into an executable directed acyclic graph (DAG) ready for human review and runtime orchestration.

4.6.1 Graph-Based Plan Representation

Execution plans are represented as directed graphs $G = (V, E, \phi)$ where:

- V is a set of task nodes, each annotated with its assigned agent (from the ICNP results), required capabilities, bid score, and metadata.
- $E \subseteq V \times V$ is a set of directed edges representing execution dependencies (task v_i must complete before task v_j begins).
- ϕ is an optional domain-specific ontology that captures additional semantic relationships and constraints among tasks and agents.

A Graph Builder Agent constructs G by incorporating the affordance markers from the bidding phase, creating a task node with the assigned agent and capability annotations for each marker, and adding dependency edges based on the task decomposition. The agent

also creates *contract award* records and *reservation markers* that formally bind agents to their assigned tasks.

4.6.2 SHACL Constraint Validation

To ensure plan correctness, SHACL constraint validation is employed [13], based on the W3C Shapes Constraint Language for validating RDF graphs [11] against structural and semantic constraints. Four categories of constraints are defined, each targeting a specific class of plan defects:

4.6.2.1 C1 - Structural Completeness

Every task node must have an assigned agent, a non-empty description, and at least one required capability specified. Formally, the shape constraint requires:

```
stig:TaskNodeShape a sh:NodeShape ;
    sh:targetClass stig:TaskNode ;
    sh:property [
        sh:path stig:assignedAgent ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
    ] ;
    sh:property [
        sh:path stig:description ;
        sh:minCount 1 ;
        sh:datatype xsd:string ;
    ] ;
    sh:property [
        sh:path stig:requiresCapability ;
        sh:minCount 1 ;
    ] .
```

4.6.2.2 C2 - Graph Integrity

All dependency edges must reference valid task nodes (no dangling references), and the graph must be acyclic. The acyclicity constraint ensures that there exists a valid topological ordering for execution. This is validated using cycle detection (e.g. Kahn's algorithm [81, 82]).

```
stig:EdgeShape a sh:NodeShape ;
    sh:targetClass stig:DependencyEdge ;
    sh:property [
        sh:path stig:fromTask ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:class stig:TaskNode ;
        sh:nodeKind sh:IRI ;
    ] ;
    sh:property [
        sh:path stig:toTask ;
        sh:minCount 1 ;
```

```
        sh:maxCount 1 ;
        sh:class stig:TaskNode ;
        sh:nodeKind sh:IRI ;
    ] .
```

The acyclicity property is verified programmatically as part of the validation pipeline (see Algorithm 3).

Algorithm 3 Plan Validation with SHACL-Inspired Constraints

Require: Execution graph $G = (V, E, \phi)$, constraint shapes $\mathcal{S}$, max iterations N
Ensure: Validated graph G^* or validation failure

for $n = 1$ to N **do**
  $report \leftarrow \emptyset$
  C1: Structural Completeness
  for each task node $v \in V$ **do**
    if $v.agent = \text{NULL}$ **or** $v.capabilities = \emptyset$ **then**
      $report$.ADD("StructuralViolation", v)
    end if
  end for
  C2: Graph Integrity
  for each edge $(v_i, v_j) \in E$ **do**
    if $v_i \notin V$ **or** $v_j \notin V$ **then**
      $report$.ADD("DanglingEdge", (v_i, v_j))
    end if
  end for
  $hasCycle \leftarrow$ DETECTCYCLE(G) ▷ Kahn's algorithm
  if $hasCycle$ **then**
    $report$.ADD("CyclicDependency", G)
  end if
  C3: Capability Alignment
  for each task node $v \in V$ **do**
    for each $c \in v.requiredCapabilities$ **do**
      if $c \notin v.agent.capabilities$ **then**
        $report$.ADD("CapabilityMismatch", v, c)
      end if
    end for
  end for
  C4: Ontology Conformance
  $report \leftarrow report \cup$ VALIDATEONTOLOGY(G, ϕ)
  if $report = \emptyset$ **then**
    return G ▷ All constraints satisfied
  else
    $G \leftarrow$ GRAPHBUILDER.REFINE($G, report$) ▷ Iterative refinement
  end if
end for
return VALIDATIONFAILURE($report$)

4.6.2.3 C3 - Capability Alignment

For each task node, the assigned agent must possess all capabilities required by the task. This constraint ensures that the ICNP assignment results are semantically valid [14]:

```
stig:CapabilityAlignmentShape a sh:NodeShape ;
    sh:targetClass stig:TaskNode ;
    sh:sparql [
        sh:message "Agent lacks required capability" ;
        sh:select """
            SELECT $this
            WHERE {
                $this stig:requiresCapability ?cap .
                $this stig:assignedAgent ?agent .
                FILTER NOT EXISTS {
                    ?agent stig:hasCapability ?cap .
                }
            }
        """ ;
    ] .
```

4.6.2.4 C4 - Ontology Conformance

Task types, tags, and inter-task relationships must conform to domain-specific ontological constraints [9, 45]. For example, a task tagged as "data-processing" must not depend on a task tagged as "user-facing" without an intermediate validation step, or certain capability combinations may be mutually exclusive within a single agent assignment.

The validation process generates a structured *validation report* documenting any constraint violations, their locations within the graph, and the specific constraint that was violated. If violations are detected, the system loops back to the Graph Builder Agent for iterative refinement, up to a configurable maximum number of iterations.

4.6.3 DAG Construction and Execution Ordering

Once a validated plan graph $G = (V, E, \phi)$ has passed all constraint checks, it must be transformed into an execution-ready DAG optimised for the runtime orchestrator. This transformation has two goals: (1) ensure that the graph is a true DAG with a well-defined topological ordering, and (2) expose structural properties, parallel task groups, fan-out/fan-in patterns, and the critical path, that guide efficient and adaptive execution.

Kahn's algorithm [81, 82] is applied to compute a topological ordering σ of V. The algorithm proceeds by iteratively removing nodes with zero in-degree. If all nodes are eventually removed, the graph is confirmed acyclic and the removal sequence defines σ. Each node v is assigned a *topological depth* $d(v) \in \mathbb{N}$ with

$$d(v) = \begin{cases} 0 & \text{if } v \text{ has no predecessors} \\ 1 + \max_{(u,v)\in E} d(u) & \text{otherwise.} \end{cases}$$

Tasks sharing the same topological depth form a parallel execution phase. Formally, phase Φ_k is defined as:

$$\Phi_k = \{v \in V \mid d(v) = k\}.$$

Tasks within the same phase have no mutual dependencies and can execute concurrently.

The *critical path* is the longest directed path from any root node to any leaf node, measured in task count (or weighted by estimated task duration). It determines the minimum possible execution time for the entire plan. Formally, for each node v, the

critical path length $cp(v)$ is:

$$cp(v) = \begin{cases} 1 & \text{if } v \text{ has no predecessors} \\ 1 + \max_{(u,v)\in E} cp(u) & \text{otherwise} \end{cases}$$

The critical path $\mathcal{C} \subseteq V$ consists of nodes on any path achieving the maximum cp value. Tasks on $\mathcal{C}$ must be prioritised during execution because delays to any critical-path task directly increase overall execution time.

Algorithm 4 DAG Construction and Execution Ordering

Require: Validated plan graph $G = (V, E)$
Ensure: Execution DAG with phases $\{\Phi_k\}$, critical path $\mathcal{C}$, and node annotations

Step 1: Topological Sort (Kahn's Algorithm)
$indegree[v] \leftarrow |\{(u, v) \in E\}|$ for all $v \in V$
$queue \leftarrow \{v \mid indegree[v] = 0\}$
$\sigma \leftarrow []$, $depth[v] \leftarrow 0$ for all $v \in V$
while $queue \neq \emptyset$ **do**
    $v \leftarrow$ DEQUEUE($queue$)
    σ.APPEND(v)
    for each $(v, w) \in E$ **do**
        $indegree[w] \leftarrow indegree[w] - 1$
        $depth[w] \leftarrow \max(depth[w],\ depth[v] + 1)$
        if $indegree[w] = 0$ **then**
            $queue$.ENQUEUE(w)
        end if
    end for
end while
if $|\sigma| \neq |V|$ **then**
    return FAIL("Cycle detected") ▷ Residual nodes indicate a cycle
end if
Step 2: Phase Assignment
for each $v \in V$ **do**
    $\Phi_{depth[v]}$.ADD(v)
end for
Step 3: Critical Path
$cp[v] \leftarrow 1$ for all $v \in V$
for each $v \in \sigma$ **do**
    for each $(v, w) \in E$ **do**
        $cp[w] \leftarrow \max(cp[w],\ cp[v] + 1)$
    end for
end for
$\mathcal{C} \leftarrow$ BACKTRACKCRITICALPATH(cp, E)
Step 4: Structural Annotation
for each $v \in V$ **do**
    Annotate v as fan-out / fan-in / root / leaf based on E
end for
return $(\sigma,\ \{\Phi_k\},\ \mathcal{C},$ annotations$)$

The DAG construction also annotates nodes with their structural roles:

- **Fan-out nodes:** v with $|\{(v, w) \in E\}| > 1$ - these trigger parallel branches and may require resource pre-allocation.

- **Fan-in (join) nodes:** v with $|\{(u, v) \in E\}| > 1$ - these synchronise multiple upstream branches and must wait for all predecessors before proceeding.
- **Root nodes:** v with $|\{(u, v) \in E\}| = 0$ - these initiate execution phases.
- **Leaf nodes:** v with $|\{(v, w) \in E\}| = 0$ - these produce final outputs.

Algorithm 4 formalises the complete DAG construction procedure. Its output, comprising execution phases, the critical path, and structural annotations, is persisted to the coordination store and passed to both the Human Validation phase and, upon approval, the runtime execution layer. This formally defined DAG serves as the authoritative execution contract for all downstream components.

4.7 Human Validation and Approval

The Human Validation and Approval phase (Step (f) in Figure 5) ensures that human oversight is applied proportionally to plan risk, fulfilling Design Goal 3 (Safety and human oversight) [44, 8]. A risk-based routing mechanism determines the approval workflow for each plan.

4.7.1 Risk-Based Routing

When a validated plan reaches this phase, the system evaluates its aggregate risk level by examining the risk classifications of all constituent tasks. A plan is routed to the *mandatory human sign-off* workflow if any of the following conditions hold:

- Any task $\tau \in G.V$ has $\tau.\mathit{risk\_level} = \text{high}$.
- Any task $\tau \in G.V$ has $\tau.\mathit{requires\_human\_approval} = \textsc{true}$.

Plans that meet neither condition proceed directly to user plan validation, where the user reviews and may refine the plan before execution (see Subsection 4.7.3). This deterministic routing ensures consistent and predictable approval workflows, where high-impact operations always receive mandatory human review while standard operations can proceed efficiently [2].

4.7.2 Human Sign-off for High-Risk Plans

For plans routed to mandatory sign-off, the system presents the complete plan details, task assignments, agent selections, dependency structure, and risk factors, to an authorised approver. The approver must explicitly approve or reject the plan. The sign-off decision is recorded as a structured audit record containing:

- Approver identity and authorisation level
- Decision timestamp
- Decision outcome (approved/rejected)
- Optional rationale text

Rejected plans are blocked from execution and may be routed back to earlier phases for revision.

4.7.3 User Plan Validation and Iterative Refinement

All plans (whether after mandatory sign-off or direct routing) are presented to users for validation. Users review the plan's graph structure, agent-task assignments, and execution flow. If modifications are needed, users can alter agent-task assignments, dependency edges, or task parameters. The system applies these modifications and routes the updated

plan back to the Graph Builder Agent for reconstruction and re-validation, creating an iterative refinement loop:

$$G_0 \xrightarrow{\text{user mods}} G_1 \xrightarrow{\text{rebuild}} G_1' \xrightarrow{\text{validate}} \begin{cases} \text{approved} & \text{if user accepts} \\ G_1' \xrightarrow{\text{user mods}} G_2 \to \cdots & \text{otherwise} \end{cases}$$

All modification decisions and resulting plan changes are logged to the audit trail, preserving a complete record of plan evolution and the rationale for each change.

4.8 Stigmergic Execution and Adaptive Strategy

The Execution phase (Steps (e) and (g) in Figure 5) transforms validated plans into running task workflows. Two components provide the key research contributions of this phase: the *Pheromone Engine* and the *LLM Strategist Agent*.

4.8.1 Pheromone Engine

The Pheromone Engine manages the stigmergic information infrastructure that underlies all coordination decisions. It maintains *affordance markers*, `stig:AffordanceMarker` instances, with intensity values $I \in [I_{\min}, I_{\max}]$ that encode the strength of agent-task associations.

The engine operates in two modes:

1. **Initialization -** The engine seeds the marker landscape with baseline intensities derived from the ICNP bidding results. A *Path Seeder* component creates preliminary markers for candidate agents based on capability alignment, establishing the initial coordination landscape before execution begins.
2. **Runtime Maintenance -** As tasks execute, the engine updates marker intensities based on feedback:

$$I'(a_i, \tau) = \begin{cases} \min(I(a_i, \tau) + \delta^+,\ I_{\max}) & \text{if } \tau \text{ succeeded with } a_i \\ \max(I(a_i, \tau) - \delta^-,\ I_{\min}) & \text{if } \tau \text{ failed with } a_i \end{cases} \tag{6}$$

where δ^+ and δ^- are reinforcement and penalty magnitudes. Concurrent decay (Eq. 1) ensures that markers not refreshed by recent activity gradually weaken.

This feedback-driven mechanism enables the system to learn from execution outcomes. Successful agent-task pairings are reinforced, while problematic associations are weakened, steering future assignments toward more reliable configurations [51].

4.8.2 LLM Strategist Agent

The LLM Strategist Agent provides adaptive, runtime replanning by observing the evolving execution state and proposing validated modifications to the plan graph.

The strategist first computes a deterministic baseline using the DAG phases and critical path produced by Algorithm 4, where execution phases group tasks that can run in parallel and the critical path identifies the longest dependency chain, which determines the minimum execution time [82].

When the strategist detects problems during execution, such as repeated failures on a critical path, emerging bottlenecks, or degraded agent performance, it formulates candidate remedies as graph deltas, which are proposed edits to the plan graph's tasks, edges, or assignments. These deltas may include:

- **Re-routing -** reassigning a failed task to an alternative agent with compatible capabilities.
- **Re-phasing -** reorganizing task phases to expose more parallelism or bypass blocked paths.
- **Bypassing -** temporarily skipping non-critical tasks to maintain progress on the critical path.

The LLM analyses the current task graph, execution history, and pheromone landscape to generate and prioritize these adaptations. Crucially, all proposed deltas are validated before application, as the system verifies that each delta preserves acyclicity, respects dependencies, and maintains exactly one assignment per task. Only validated deltas are committed to the coordination graph store.

The resulting execution strategy evolves over time and comprises a textual summary of the execution approach, organised phases for parallel and sequential execution, the critical path, and any applied graph deltas together with their justifications. This dynamic strategy guides the orchestration of task launches and provides full visibility into both the initial plan and its runtime adjustments.

Algorithm 5 outlines the adaptive strategy process.

Algorithm 5 LLM Strategist: Adaptive Runtime Replanning

Require: Execution graph $G = (V, E)$, execution state σ
Ensure: Updated strategy Π and (optionally) modified graph G'

$phases \leftarrow \textsc{TopologicalSort}(G)$ ▷ Group parallel tasks into phases
$critPath \leftarrow \textsc{CriticalPath}(G)$ ▷ Longest dependency chain
$\Pi \leftarrow \textsc{BuildBaselineStrategy}(phases, critPath)$
while execution is active **do**
  $\sigma \leftarrow \textsc{ObserveExecutionState}$ ▷ Task outcomes, delays, failures
  if σ contains failures or anomalies **then**
    $\Delta \leftarrow \text{LLM.}\textsc{ProposeDeltas}(G, \sigma, critPath)$
    for each delta $\delta \in \Delta$ **do**
      $G' \leftarrow \textsc{ApplyDelta}(G, \delta)$
      if $\textsc{IsAcyclic}(G')$ **and** $\textsc{AllTasksAssigned}(G')$ **then**
        $G \leftarrow G'$ ▷ Accept validated delta
        $\Pi \leftarrow \textsc{UpdateStrategy}(\Pi, \delta)$
        $\textsc{LogToAudit}(\delta,$ "accepted")
      else
        $\textsc{LogToAudit}(\delta,$ "rejected: constraint violation")
      end if
    end for
  end if
end while
return Π, G

4.8.3 Ethics-Check Gate

Before execution, an Ethics-Check Agent performs a final safety review. The agent receives the complete plan context and uses LLM-based reasoning to identify potential ethical concerns, safety issues, or policy violations [2]. Unlike domain-specific validation agents (e.g. a compliance agent that enforces business rules for a particular industry), the Ethics-Check Agent is domain-agnostic. It enforces universal safety guarantees that apply

regardless of the application domain. Concretely, the Ethics-Check Agent post-validates the outputs of all other agents in the pipeline and enforces a set of high-level guarantees:

- **No hallucinated data -** the agent flags outputs that make unverifiable factual claims or that contradict information already present in the plan context.
- **No unfair bias -** the agent checks for discriminatory patterns in decisions or recommendations against protected groups.
- **No sensitive-data leakage -** the agent inspects outputs for personal data, confidential business information, or other sensitive content that should not leave the system boundary.
- **No unsafe approvals -** before authorising any high-risk action, the agent verifies that it has passed the required human-approval and risk-based checks [8].

If any of these checks fail, the Ethics-Check Agent can veto execution or request re-evaluation, for example by sending the plan back to the LLM Strategist (Subsection 4.8.2) or to a human reviewer. All ethics decisions, including vetoes and manual overrides, are recorded in the audit trail along with their underlying rationales. This enables post-hoc review of the framework's ethical evaluation process.

4.9 Failure Handling and Escalation

The framework treats failures not as terminal events but as signals that trigger a structured, multi-tier recovery process. This design fulfills Design Goal 5 (Adaptive failure recovery and strategic replanning).

4.9.1 Tier 1: Bounded Local Retries

At each critical execution point, operations are wrapped with configurable retry policies specifying maximum attempts, backoff curves (e.g. exponential), and abort-vs-degrade behaviour [54, 61]. This tier absorbs transient faults such as network glitches, temporary service unavailability, and intermittent timeouts, without escalation. When retries succeed within bounds, the failure remains observable to the rest of the system.

4.9.2 Tier 2: Adaptive Replanning

When local retries are exhausted and the failure remains unresolved, the LLM Strategist (Section 4.8) treats them as signals to reconsider the execution strategy, before escalating further. The strategist may propose graph deltas such as re-routing failed tasks to alternative agents or re-phasing execution to bypass bottlenecks. Simultaneously, the Pheromone Engine reduces marker intensities for problematic agent-task pairings (Eq. 6), steering future assignments away from unreliable configurations.

4.9.3 Tier 3: Structured Escalation

When failures exhaust the configured retry budget and adaptive replanning has not resolved the issue, the system raises a structured escalation event. The escalation record captures:

- Run and plan identifiers
- Machine-readable failure reason (e.g. "retries exhausted")
- Severity level (derived from the plan's risk classification)
- Recent execution summaries (per-task outcomes, success rates, timing)
- Indicator that automated recovery has been exhausted

These records are consumed by monitoring dashboards, alerting systems, or higher-level agents that coordinate human intervention. The escalation payload provides sufficient context for human operators to quickly assess the situation and decide on corrective actions.

4.9.4 Knowledge Capture and Learning

After human-assisted resolution, the system persists a structured description of the failure, its root cause, and the applied remediation into a knowledge base. Over time, this knowledge base accumulates failure patterns that the system can leverage in future runs, for example by penalising agents that frequently participate in escalated failures during the bidding phase, or by enriching the LLM Strategist's context with historical failure patterns when proposing runtime adaptations.

Algorithm 6 summarises the multi-tier failure handling process.

Algorithm 6 Multi-Tier Failure Handling and Escalation

Require: Failed task τ, execution context σ, retry config ($maxRetries$, $backoff$)
Ensure: Recovery outcome or escalation record

Tier 1: Bounded Local Retries
for $k \leftarrow 1$ **to** $maxRetries$ **do**
  WAIT($backoff(k)$)
  $result \leftarrow$ RETRYTASK(τ)
  if $result =$ SUCCESS **then**
    PHEROMONEENGINE.REINFORCE($\tau.agent$, τ)
    return RECOVERED
  end if
end for
Tier 2: Adaptive Replanning
$\Delta \leftarrow$ STRATEGIST.PROPOSEADAPTATION(τ, σ)
if $\Delta \neq \emptyset$ **and** APPLYANDVALIDATE(Δ) **then**
  PHEROMONEENGINE.PENALIZE($\tau.agent$, τ)
  return REPLANNED
end if
Tier 3: Structured Escalation
$record \leftarrow$ CREATEESCALATIONRECORD(τ, σ, `"retries exhausted"`)
NOTIFYHUMANOPERATOR($record$)
PERSISTTOKNOWLEDGEBASE($record$, $\tau.rootCause$)
return ESCALATED($record$)

4.10 Audit and Traceability

The Audit and Traceability layer is a cross-cutting concern that spans all phases of the framework, fulfilling Design Goal 4 (End-to-end audit traceability). Rather than a standalone phase, it is an integral fabric that ties all system events into coherent, queryable narratives.

4.10.1 End-to-End Correlation Model

The system assigns stable correlation identifiers [83] to each pipeline and run, and propagates them through all phases and components. Every event, including task intake, bidding outcomes, plan generation, human approvals, ethics checks, execution outcomes, failure escalations, and pheromone updates, is annotated with these identifiers. This enables reconstruction of complete task lifecycles, tracing each pipeline from the initial user request through validation, bidding, planning, approval, execution, and resolution.

4.10.2 Structured Audit Records

All critical decisions and state transitions are captured as structured records (not unstructured text logs). Each record contains at minimum what happened, when it occurred, which pipeline or run it relates to, and which component initiated the change. Where relevant, records also capture input/output summaries while avoiding storage of sensitive raw content. Because all subsystems write into a common audit stream using a shared schema, downstream tools can query, filter, and aggregate audit data across phases.

4.10.3 Traceability Scoring

Building on the unified audit stream, the system computes *traceability scores* that quantify governance coverage for each run. Runs that have undergone validated planning, human approval, ethics review, completed execution, and resolved escalation (if applicable) receive higher scores. These metrics enable policy-driven requirements (e.g. enforcing minimum traceability scores for regulated environments [8, 2]) and highlight areas where additional controls may be needed.

4.10.4 Supporting Explanation and Continuous Learning

The accumulated audit data supports two forward-looking capabilities:

1. **Decision Explanation -** when the LLM Strategist proposes a runtime adaptation, the system can link that decision to the specific failures, human interventions, or escalation records that motivated it, providing rich explanations of system behaviour [84].
2. **Continuous Learning -** over time, trace data feeds into the system's knowledge base, informing updates to validation rules, bidding heuristics, runtime adaptation policies, and failure handling strategies. The audit layer thus actively enables system improvement, not just passive record-keeping.

4.11 Summary

This chapter has presented the design of the proposed Multi-Agent coordination framework as an integrated response to the three architectural gaps identified earlier. Five design goals, namely decentralised coordination, formal ontological grounding, embedded safety, end-to-end traceability, and adaptive resilience, established the criteria against which every architectural choice was justified. The framework was then described as a six-step pipeline spanning agent registration, task normalisation, stigmergic task assignment through ICNP, plan generation and SHACL validation, risk-based human approval, and adaptive execution, with audit and traceability operating as a cross-cutting fabric throughout. The Pheromone Engine, the LLM Strategist, and the Ethics-Check Gate

together provide the adaptive learning, runtime replanning, and domain-agnostic safety guarantees that distinguish this framework from current orchestration platforms, while a multi-tier failure handling mechanism delivers self-healing behaviour without mandatory human intervention.

In the next chapter, the focus shifts from design to realisation, describing how each conceptual component is instantiated in a cloud-native prototype and how the technology choices and infrastructure of the implementation support the architectural properties claimed here.

5. Prototype Development

This chapter describes the implementation of the Multi-Agent coordination framework proposed in Chapter 4. The prototype translates the conceptual model, task normalisation, ICNP bidding, constraint-validated plan graphs, DAG-based execution, human approval workflows, adaptive execution, and failure handling, into a working system built on cloud-native infrastructure. This chapter details the technology choices, component implementations, data flows, and storage strategies that realise each element of the proposed framework. Table 2 provides a high-level mapping between the framework's theoretical components and their prototype counterparts.

Framework Component	**Prototype Module**
Agent Registration & Capability Model	Agent Registry Store, Agent Seeder Service
Task Normalisation (3-stage)	Task Intake Pipeline, Supervisor Agent
ICNP Bidding	Blueprint Bidder Agent, CFP Publisher
Plan Generation & Constraint Validation	Graph Builder Agent, Plan Validator
DAG Construction & Execution Ordering	DAG Relation Builder
Human Validation & Approval	Approval Router, Plan Review UI
Pheromone Engine	Pheromone Engine Service (sidecar)
LLM Strategist	LLM Strategist Agent
Ethics-Check Gate	Ethics-Check Agent
Failure Handling & Escalation	Retry Middleware, Escalation Agent
Audit & Traceability	Audit Logger, Correlation Middleware, OpenTelemetry

Table 2: Mapping between proposed framework components (Chapter 4) and prototype implementation modules.

5.1 Technology Stack and Cloud-Native Infrastructure

The prototype is built as an event-driven, microservice-oriented system [52] deployed as containerised services under helm charts, with cloud-native deployability as a design constraint. Each logical component runs as an independent containerised service, communicating exclusively through the message broker and the shared document store. Table 3 summarises the key technology choices.

5.1.1 Message Broker: RabbitMQ

All inter-service communication is mediated by RabbitMQ [53, 54], an AMQP-based message broker. Components publish typed messages (e.g. `start-task`, `task-completed`, `escalation-raised`, `cfp-issued`) to named exchanges, while consumers subscribe to queues bound to the relevant routing keys. The `aio-pika` client provides non-blocking publish and consume operations within each service's asyncio event loop. RabbitMQ

decouples producers from consumers, allows per-phase horizontal scaling, and provides durable queues so that messages survive broker restarts without data loss [54].

Concern	Technology	Role
API layer	FastAPI 0.111 + Uvicorn	REST endpoints for task submission and UI
Async runtime	Python 3.11 + asyncio	Non-blocking event loop for all agent services
Message broker	RabbitMQ 3 (AMQP)	Async inter-service communication via aio-pika
Document store	Couchbase	Graph marker store, agent registry, plan cache
Triple store	Apache Jena Fuseki 5.1	SPARQL/RDF knowledge graph: ontology, SHACL shapes, plan and execution graphs
Service registry	HashiCorp Consul	Service discovery and health-check integration
LLM providers	OpenAI API (GPT-4o family)	Supervisor, Strategist, and Ethics-Check agents
Data validation	Pydantic v2	Schema enforcement for all data models
Retry logic	tenacity	Configurable retry/backoff policies
Observability	OpenTelemetry SDK + OTLP	Distributed tracing, metrics, and structured logs
Containerisation	Docker + Helm	Service isolation and local deployment

Table 3: Prototype technology stack.

5.1.2 Document Store

Couchbase [85] serves as the document-backed coordination store that persists all operational artifacts produced by the coordination layer, including `stig:TaskSpec` objects, `stig:AffordanceMarker` instances, agent registry entries, plan drafts, execution maps, and audit records. Documents are organised into named collections, one per entity type, enabling efficient access by key, collection scan, or N1QL [7] for complex lookups. Couchbase's upsert semantics directly support the idempotent write patterns required for at-least-once message delivery [60]. The semantic layer is kept separate, with RDF triples, ontology axioms, and SHACL shapes stored in the dedicated triple store described in Subsection 5.1.3.

5.1.3 Triple Store

Fuseki serves as the semantic layer's persistence and query substrate, providing the native RDF triple store kept separate from the coordination store. The deployed instance is Apache Jena Fuseki 5.1 [86], which exposes a SPARQL 1.1 Query and Update endpoint together with the SPARQL 1.1 Graph Store Protocol over HTTP. A single dataset named `agentcy` hosts a set of named graphs that partition the system's knowledge by role, where `<.../graphs/ontology>` holds the OWL ontology defined in Chapter 4, `<.../graphs/shapes>` holds the SHACL shapes used by the Plan Validator (Section 5.5), and

per-plan graphs hold the plan, execution, data-flow, provenance, and domain-knowledge triples emitted at runtime. Ontology and shapes are registered idempotently at startup via the Graph Store Protocol, so repeated deployments converge on the same schema state without duplicating triples. Downstream services query Fuseki via SPARQL for reasoning that is awkward in a document model, including transitive capability lookups over the taxonomy, cross-plan retrieval of similar executions for the plan recommender, and PROV-O traversals over `prov:wasGeneratedBy` and `prov:used` chains for audit.

5.1.4 Service Registry

HashiCorp Consul [87] provides service discovery and health checking for all agent microservices. When a new agent service starts, it registers itself with Consul, advertising its name, network address, and capability metadata. The Agent Seeder Service queries Consul to discover registered services and automatically populate the agent registry in Couchbase. Consul health checks monitor each service's liveness, and the Heartbeat Monitor (Section 5.2) uses Consul status events to trigger pheromone decay on inactive agents.

5.1.5 LLM Integration

LLM capabilities are provided through a configurable `LLM_Connector` abstraction built on the OpenAI Python SDK. Model names and prompt templates are externalised to environment configuration, allowing the three LLM-driven agents, namely the Supervisor, LLM Strategist, and Ethics-Check Agent, to be independently configured.

5.1.6 Observability

All services are instrumented with the OpenTelemetry Python SDK [88]. Distributed traces are propagated across service boundaries via message headers [83], enabling end-to-end tracing of a task through the full pipeline. Traces and metrics are exported via OTLP to a configurable backend, using Jaeger for traces and Prometheus for metrics. This provides the runtime observability foundation required by the Audit and Traceability layer described in Section 5.10.

5.2 Agent Registration Implementation

The Agent Registration component, implementing Section 4.3, consists of three subcomponents, namely the Agent Registry Store, the Agent Seeder Service, and the Heartbeat Monitor.

5.2.1 Agent Registry Store

The Agent Registry Store persists agent profiles as JSON documents in the `agent_registry` Couchbase collection, implementing the `stig:Agent` ontology class described in Section 4.3. Each profile includes a unique agent URI derived from the agent's Consul service name, a validated list of `stig:Capability` instances, stigmergy tag intensities with decay parameters and last-update timestamps, and status fields covering `active`, `inactive`, and `degraded` states together with correlation metadata. The store exposes typed CRUD operations supporting queries by capability, status, and minimum tag intensity, and uses Couchbase upsert semantics to guarantee idempotency under at-least-once message delivery.

5.2.2 Agent Seeder Service

The Agent Seeder automates the transformation of Consul service registrations into agent registry entries. When a seeding request is received via RabbitMQ, the service:

1. Queries Consul for the requested service entries (or all services if unspecified).
2. For each service, resolves its capability list (from default capability mappings and per-service overrides), assigns an agent URI, initialises stigmergy tags from global policy, and sets status to `active`.
3. Persists the resulting `AgentRegistryEntry` to Couchbase via an upsert, handling both first-time registration and re-seeding idempotently.

This automated seeding ensures that all deployed services are immediately available as agents without manual registration, maintaining consistency between Consul and the agent registry.

5.2.3 Heartbeat and Availability Monitoring

Agent services emit periodic heartbeat messages to RabbitMQ. The Heartbeat Monitor subscribes to the heartbeat queue, updates the last-seen timestamp in the agent's Couchbase document, and marks agents as `degraded` when the elapsed time since the last received heartbeat exceeds a configurable staleness threshold, and as `inactive` when it exceeds a higher offline threshold. When agents are marked inactive, the Pheromone Engine applies Eq. 1 to their stigmergy tags, reducing their influence on future bidding decisions.

5.3 Task Input Pipeline Implementation

The Task Input Pipeline implements the three-stage normalisation procedure defined in Section 4.4. The pipeline is exposed as a FastAPI endpoint and structured as a sequence of middleware components.

5.3.1 API Endpoint and Message Format

Tasks are submitted via a FastAPI [89] POST endpoint that accepts JSON payloads in three formats, namely a direct `TaskSpec` JSON object, a natural language description wrapped in a message envelope, and hybrid payloads containing both structured and unstructured content. Pydantic [90] v2 models enforce the envelope schema at ingestion, and invalid payloads are rejected with a structured error response before any pipeline processing occurs.

5.3.2 Input Validation and Content Filtering

The validation middleware implements Stages 1 and 2 of Algorithm 1. It reads `pipeline_id` and `pipeline_run_id` from the message envelope and rejects payloads with missing fields, then inspects a `content_filter_passed/blocked` flag set by an upstream content filter service and rejects blocked payloads with a policy-violation error. The content filter is implemented as a pluggable Pydantic-validated component, allowing organisations to substitute domain-specific logic. Surviving payloads are normalised from various input shapes (dict objects, typed Pydantic models, nested arrays) into a canonical internal representation, with multiple task specifications in a single payload processed individually.

5.3.3 LLM-Based Structuring

The Supervisor Agent implements Stage 3 of Algorithm 1 through the `LLM_Connector` abstraction. It constructs a structured prompt that includes the Pydantic `TaskSpec` schema as context, sends it to the configured provider (OpenAI GPT-4o), and validates the response against the Pydantic model. On validation failure, the agent falls back to a rule-based heuristic that maps known payload fields to schema attributes. The finalised `TaskSpec` is persisted to the `task_specs` Couchbase collection.

5.4 Blueprint Bidding Implementation

The Blueprint Bidding module realises the ICNP mechanism defined in Section 4.5 and Algorithm 2. It consists of a CFP Publisher, per-agent Evaluators, and a Bid Selector.

5.4.1 Call for Proposals (CFP) Publication

For each validated `TaskSpec`, the Blueprint Bidder constructs a CFP message and publishes it to the `cfp` RabbitMQ exchange, from which all registered agent services receive it. The system maintains a per-task state machine in Couchbase: `cfp_issued` → `bids_received` → `awarded` (or `cfp_reissued` if the TTL expires without bids). On TTL expiry, the CFP is reissued with amplified stimulus $s_{k+1} = s_k \cdot (1 + \alpha)$, up to K rounds as specified in Algorithm 2.

5.4.2 Agent Evaluation

Each agent service subscribes to the `cfp` queue and evaluates incoming CFPs by computing $S(a_i, \tau)$ and comparing it against θ_i as defined in Section 4.5. The implementation retrieves the required inputs from Couchbase, including the agent's capability set, current task load, trust score (computed as the success rate over a configurable sliding window of past executions), and dynamic threshold value. Capability match is computed directly from the two capability sets. Bid decisions are made locally by each agent service with no central coordinator.

5.4.3 Bid Generation and Selection

Agents whose stimulus score exceeds their threshold publish a bid message to the `bids` RabbitMQ exchange, containing the bid score $B(a_i, \tau)$ (Eq. 5), agent identifier, and supporting metadata including capability match breakdown, trust evidence, and cost estimate. The Blueprint Bidder collects bids within the TTL window, ranks them by score, and selects the top bid. The winning bid is materialised as a `stig:AffordanceMarker` document in the `markers` Couchbase collection with its intensity initialised to the bid score, and non-winning bids are persisted to the audit collection.

The scoring weights $\gamma_1, \gamma_2, \gamma_3$ in Eq. 5 are exposed as configuration parameters and set in the prototype to $\gamma_1 = 0.25$ for capability match and $\gamma_2 = \gamma_3 = 0.375$ for trust and cost-effectiveness. Capability match receives a lower weight because structural incompatibility is already enforced upstream by the eligibility check at line 9 of Algorithm 2, leaving γ_1 to act as a fit signal rather than a hard filter among admitted bidders. Trust and cost-effectiveness are weighted symmetrically as they encode complementary signals of equal importance, where trust summarises historical reliability and $1 - cost/maxCost$ captures prospective resource expenditure.

5.4.4 Assignment Freezing

Once an affordance marker is written to Couchbase, a `frozen` status flag is set on the document, preventing any automated process from modifying the assignment. Only an explicit human action through the plan review interface (Section 5.7) or a system-level override operation can unfreeze and reassign. This implements the planning stability property described in Section 4.5.

5.5 Plan Generation and Validation Implementation

The Plan Generation implementation translates bidding results into an executable plan and validates it against the constraint categories defined in Section 4.6.

5.5.1 Graph Builder Agent

The Graph Builder Agent reads frozen affordance markers and original `TaskSpec` documents from Couchbase and constructs the execution graph $G = (V, E, \phi)$:

1. For each marker, creates a task node annotated with the assigned agent, required capabilities, bid score, and task metadata.
2. Infers dependency edges from the task specifications and any domain ontology declared in the pipeline configuration.
3. Creates *contract award* documents in Couchbase that formally bind agents to their assignments, referencing the originating bid.
4. Creates *reservation markers* that prevent the assigned agent from being double-booked by concurrent pipelines.

The resulting graph specification is written as a Pydantic-validated JSON document (`PlanDraft`) to the `plan_drafts` Couchbase collection.

5.5.2 Plan Validator

The Plan Validator implements constraints C1–C4 from Section 4.6 as a Python service. SHACL constraints are encoded as a Turtle shape file loaded into the Fuseki triple store at startup via the Graph Store Protocol (Subsection 5.1.3), and plan drafts are validated against the `<.../graphs/shapes>` named graph using the `pyshacl` reasoner. A complementary programmatic validator enforces the same constraints on the raw graph specification, primarily to provide structured violation reports without round-tripping through SPARQL. Violations from both checkers are consolidated into a single validation report attached to the `PlanDraft` document, which is then either marked `final_conforms = True` or handed back for iterative refinement as defined in Section 4.6, capped at a configurable iteration budget.

5.5.3 Plan Cache

The Plan Cache stores validated `PlanDraft` documents in Couchbase for potential reuse. Cache keys are derived from a SHA-256 fingerprint of the graph specification (task identifiers, agent assignments, and dependency edges). On a cache hit, the system verifies that the cached plan's agent assignments are still valid (agents still active, capabilities unchanged) before serving it. Cache hits, misses, and invalidations are logged to the audit stream.

5.5.4 Map Generation Module

The Map Generation Module transforms the validated plan into an execution map consumed by the runtime orchestrator:

1. Produces ordered task execution sequences respecting topological dependency order.
2. Resolves each assigned agent's network endpoint from Consul, translating agent URIs to service addresses.
3. Generates execution metadata per task: runtime environment parameters, artifact references, and credential identifiers.
4. Produces a fallback execution map with alternative agent assignments for contingency scenarios.

The execution map is written to the `execution_maps` Couchbase collection and serves as input to both the Path Setup and Human Validation phases.

5.6 Path Setup Implementation

The Path Setup phase establishes the stigmergic infrastructure and the formal execution DAG that guide runtime coordination.

5.6.1 Pheromone Engine Service

The Pheromone Engine runs as a sidecar microservice [52] that subscribes to task-outcome events on RabbitMQ and manages affordance marker intensities in Couchbase. On pipeline start, it seeds markers from the ICNP results and applies the decay function (Eq. 1) to any pre-existing markers. On each `task-completed` or `task-failed` event, it applies Eq. 6 and clamps intensities to $[I_{\min}, I_{\max}]$. An asyncio background task applies decay at a configurable interval to ensure the pheromone landscape reflects recent activity. All intensity updates are written to Couchbase with full audit metadata and emitted as OpenTelemetry span events for observability.

5.6.2 Path Seeder

The Path Seeder runs in parallel with task validation, establishing preliminary affordance markers before formal bidding begins. For each incoming `TaskSpec`, it queries the Couchbase agent registry for agents with matching capabilities, ranks them by capability alignment and current tag intensity, and writes preliminary `stig:AffordanceMarker` documents with intensities proportional to suitability scores. These seeded markers bias the ICNP process toward well-qualified agents. On a fresh registry, the `stig:Stigma` tag intensities used for ranking are initialised from a capability-policy default declared at agent-seeding time and only grow from that baseline as committed markers accumulate through successful runs.

5.6.3 DAG Relation Builder

The DAG Relation Builder applies Algorithm 4 to the validated `PlanDraft` read from Couchbase, executing Kahn's topological sort over a dict-based adjacency representation, grouping tasks into parallel phases by topological depth, and computing the critical path through longest-path analysis. The implementation adds a connectivity check beyond the algorithm specification, requiring every task node to be reachable from at least one root node, since disconnected subgraphs would produce orphaned tasks at runtime. Nodes

are annotated with their structural roles (root, leaf, fan-out, fan-in) and the resulting DAG document is persisted to Couchbase, where it is consumed by the LLM Strategist at execution kickoff.

5.7 Human Validation Implementation

The Human Validation implementation realises the risk-based approval routing defined in Section 4.7.

5.7.1 Approval Router

The Approval Router applies the risk-based routing rule defined in Section 4.7 to the validated `PlanDraft` read from Couchbase, forwarding triggering plans to the Human Sign-off Service and all others directly to the User Plan Validation interface. The routing decision and its triggering tasks are written to the Couchbase audit collection.

5.7.2 Human Sign-off Service

For high-risk plans, the FastAPI-backed sign-off service presents the plan summary (task assignments, agent details, DAG structure, and risk factors) to the designated approver. The approver submits an approve/reject decision along with an optional rationale. The decision is persisted as a structured audit event in Couchbase and, on rejection, a `plan-rejected` message is published to RabbitMQ to notify the pipeline controller.

5.7.3 Plan Review Interface

The plan review interface (served by FastAPI) enables users to inspect the full plan graph, modify agent assignments, adjust dependency edges, or update task parameters. When modifications are submitted, the system writes the changes to the `PlanDraft` in Couchbase and publishes a `plan-rebuild-requested` message to RabbitMQ, triggering the Graph Builder Agent and Plan Validator to reconstruct and re-validate the plan, implementing the iterative refinement loop defined in Section 4.7. All user actions are captured in the Couchbase audit collection.

5.8 Execution Layer Implementation

The Execution Layer implements the runtime orchestration, adaptive strategy, and safety review mechanisms described in Section 4.8.

5.8.1 Kickoff Signal and Execution Initiation

Execution is initiated by a `kickoff` message published to RabbitMQ (from a user request or an upstream pipeline event). The Execution Orchestrator consumes this message and:

1. Verifies that the `PlanDraft` in Couchbase has `is_valid = true` and any required human approval is recorded.
2. Loads the execution map and DAG document from Couchbase.
3. Initialises the execution context (run identifier, environment variables, correlation identifiers).

4. Signals the LLM Strategist by publishing a `strategy-requested` message to RabbitMQ.

5.8.2 LLM Strategist Agent Implementation

The LLM Strategist is implemented as a stateful asyncio service that maintains the current execution strategy in memory and persists strategy updates to Couchbase:

1. On initialization, the agent reads the DAG document (phases and critical path produced by Algorithm 4) from Couchbase. This deterministic baseline requires no LLM call.
2. The agent subscribes to `task-completed` and `task-failed` events on RabbitMQ, maintaining an in-memory summary of per-task outcomes, timing, and failure counts.
3. When anomalies are detected, the agent constructs a prompt for the configured LLM provider (via `LLM_Connector`) that includes the serialised graph, execution history, and critical path. The LLM returns candidate graph deltas as a structured JSON response.
4. Each delta is applied to an in-memory copy of the graph and validated against three constraints, Kahn's acyclicity check, dependency preservation, and the exactly-one-assignment rule. Only validated deltas are written to the `plan_drafts` Couchbase collection.
5. Accepted deltas are appended to the strategy history (persisted in Couchbase), with timestamps, justifications, and OpenTelemetry span links.

5.8.3 Ethics-Check Agent Implementation

The Ethics-Check Agent is triggered by a `ethics-check-requested` RabbitMQ message. It reads the complete plan context from Couchbase (task specifications, agent assignments, DAG structure, human approval records) and constructs a structured prompt for the configured LLM. The LLM returns a JSON object with an `approved` flag, a list of identified concerns, and explanatory notes. If the check fails, the agent publishes an `execution-vetoed` message to RabbitMQ and persists a veto record to the Couchbase audit collection. All outcomes (approval or veto with rationale) are stored as structured audit documents.

5.8.4 Task Execution Orchestrator

The Task Execution Orchestrator launches tasks in phase order as defined by the DAG:

1. For each task in the current phase, resolves the assigned agent's service endpoint from Consul.
2. Prepares a task payload (runtime configuration, artifact references, environment variables) and publishes a `start-task` message to RabbitMQ [53].
3. Supports multiple execution modes, including dispatching to containerised microservices via service URIs, invoking Python plugin entry points, and triggering artifact deployment jobs.
4. Subscribes to `task-completed` and `task-failed` events, aggregating per-task outcomes (status, duration, error details) into an execution report stored in Couchbase.
5. On failure, publishes a `task-failed` event to RabbitMQ, triggering both the Retry Middleware and the LLM Strategist for potential replanning.

5.9 Failure Handling and Escalation Implementation

The Failure Handling implementation realises the three-tier recovery mechanism specified in Section 4.9 and Algorithm 6.

5.9.1 Retry Middleware

Configurable retry policies are implemented using the `tenacity` library, applied as decorators at three critical execution points: agent business logic invocation, Couchbase write operations, and RabbitMQ message publication. Each policy specifies:

- **Maximum retries:** configurable per pipeline (default: 3).
- **Backoff strategy:** `tenacity` supports fixed, linear, and exponential backoff with configurable base delay and random jitter [54].
- **Abort vs. degrade:** configurable per task; some tasks abort the run on exhaustion, while others allow degraded continuation.

5.9.2 Adaptive Replanning via LLM Strategist (Tier 2)

When the retry middleware exhausts its budget, it publishes a `retries-exhausted` RabbitMQ message that the LLM Strategist service (Section 4.8.2) consumes. The Strategist invokes `LLM_Connector` with its adaptation prompt, validates each returned delta as specified in Section 4.8.2, and persists accepted deltas to the `plan_drafts` Couchbase collection. In parallel, the Pheromone Engine applies the penalty update (Eq. 6) to the failed $(\text{task}, \text{agent})$ pair. The run is marked `replanned` if at least one delta is validated, otherwise the failure is handed to Tier 3.

5.9.3 Failure Escalation Agent (Tier 3)

When both the retry middleware and the Strategist's adaptive replanning fail to resolve a failure, the Strategist publishes an `adaptive-replanning-failed` RabbitMQ message that triggers the Failure Escalation Agent. The agent reads the run's execution summaries from Couchbase, compares the attempt count against the configured escalation threshold, and if exceeded:

1. Creates a structured escalation document in Couchbase containing the run/plan identifiers, failure reason (`retries_exhausted`), severity (derived from the plan's risk classification), recent per-task outcomes, and an `automated_recovery_exhausted` flag.
2. Publishes an `escalation-raised` event to RabbitMQ, which feeds monitoring dashboards and alert integrations.

5.9.4 Human Notification and Knowledge Capture

Escalation events are consumed by a notification service that dispatches alerts through configurable channels (dashboard push notifications, email, or on-call webhooks) with the full escalation payload. After human-assisted resolution, an operator submits a resolution record via the FastAPI interface, including the root cause and applied remediation. This record is persisted to the `knowledge_base` Couchbase collection, where it accumulates over time and can be queried by the LLM Strategist when constructing adaptation prompts for future runs.

5.10 Audit and Traceability Implementation

The Audit and Traceability implementation provides the cross-cutting observability infrastructure specified in Section 4.10.

5.10.1 Correlation Middleware

A FastAPI middleware component and an `aio-pika` message interceptor automatically inject `pipeline_id` and `pipeline_run_id` into all outgoing HTTP responses, RabbitMQ message headers, Couchbase document metadata, and OpenTelemetry span attributes. This ensures every event in the system carries its originating pipeline context, enabling complete lifecycle reconstruction for any run.

5.10.2 Structured Audit Logger

The Audit Logger exposes a Python interface used by all services to write structured audit records to the `audit_log` Couchbase collection. Each record contains:

- **Event type -** categorised by pipeline phase (task-intake, bidding, plan-generation, approval, ethics-check, execution, failure, pheromone-update).
- **Timestamp -** ISO 8601 UTC timestamp.
- **Correlation IDs -** `pipeline_id` and `pipeline_run_id`.
- **Component -** the service name that generated the event.
- **Payload -** structured summary of inputs and outputs (PII and sensitive content are excluded).
- **Outcome -** result of the operation (success, failure, violation, approval, rejection, veto).

The shared Couchbase collection and Pydantic schema ensure that all audit records are uniformly queryable via N1QL across all services and phases.

5.10.3 Traceability Metrics

The system computes a traceability score for each run by evaluating which governance checkpoints were exercised, as defined in Section 4.10:

- Plan validated (constraint checks passed): +1 point
- Human approval recorded (if required by risk routing): +1 point
- Ethics review completed: +1 point
- Execution report generated: +1 point
- Escalation resolved (if applicable): +1 point

The total score, normalised to $[0, 1]$, is persisted to the run's Couchbase document and exposed via the FastAPI monitoring endpoint. Minimum score thresholds are configurable per deployment environment.

5.10.4 Integration with External Observability Systems

OpenTelemetry spans and metrics collected throughout the pipeline are exported via OTLP to pluggable backends [83]. In the reference deployment, Jaeger receives distributed traces and Prometheus scrapes service metrics. The Couchbase audit stream can additionally be tailed and forwarded to a SIEM or log aggregation system (e.g. the Elastic Stack) through a configurable exporter service [91]. Because all audit records share a common Pydantic-validated schema, downstream tooling consumes them without further

transformation, and the audit data is interoperable across tooling boundaries. The audit layer also supports event replay for debugging and post-incident analysis, where given a run's correlation identifier the system can reconstruct the complete sequence of events from task intake through execution and resolution.

5.11 Summary

This chapter has described the cloud-native realisation of the framework presented in Chapter 4. Each conceptual component has been mapped onto a concrete prototype module, with the technology stack chosen to support per-run isolation, idempotent message handling, and end-to-end traceability as structural properties rather than retrofitted features. RabbitMQ mediates all inter-service communication, Couchbase persists operational artifacts as document collections, and Apache Jena Fuseki holds the ontology, SHACL shapes, and per-plan graphs that underpin the semantic layer. Around this substrate, the agent registration, task intake, ICNP bidding, plan validation, DAG construction, human approval, and execution components have been implemented as independent microservices coordinated through typed messages and shared coordination state. Cross-cutting concerns, including correlation propagation, structured audit logging, and OpenTelemetry-based observability, have been integrated into every service to support the audit and traceability guarantees claimed in the design.

In the next chapter, the focus shifts from how the framework is built to how it actually behaves, presenting the empirical evaluation that quantifies the contribution of each coordination mechanism, validates the framework's design goals against measured outcomes, and reports on user perceptions of the resulting system.

6. Experimentation and Evaluation

This chapter evaluates the proposed Multi-Agent coordination framework through a combination of quantitative experimentation and a qualitative user perception survey. The evaluation is designed to address the central research question introduced in Chapter 1, which is operationalised through five hypotheses. Each hypothesis isolates the contribution of a specific coordination mechanism, or combination of mechanisms, and assesses its marginal effect on overall system quality.

H1 concerns stigmergic learning. *Claim: removing the pheromone engine will degrade assignment accuracy over successive runs relative to the full framework, while leaving single-run accuracy unaffected.* Under this hypothesis, the full framework is expected to exhibit a monotonically increasing accuracy trend that converges within fewer than 15 runs, whereas the pheromone-disabled configuration should remain approximately flat over time. This expectation is grounded in stigmergy theory, which predicts that agents coordinating through environmental markers can achieve emergent global optimisation without direct communication, as successful actions reinforce environmental signals that bias future decisions.

H2 addresses competitive selection. *Claim: removing ICNP bidding will produce the largest single-component degradation in assignment accuracy, and a round-robin allocation policy will yield near-random accuracy when multiple agents share the same capabilities.* This hypothesis is motivated by the Contract Net Protocol and task allocation theory, both of which predict that market-based coordination outperforms static allocation policies when agent capabilities are non-uniform.

H3 evaluates the framework's defence-in-depth approach. *Claim: the presence of both validation layers, SHACL structural validation and domain-level compliance checking, jointly reduces downstream error propagation more than either layer alone, producing a safety improvement greater than the sum of its parts.* SHACL enforces the structural well-formedness of generated plans, while the compliance layer enforces domain-specific business rules. The framework's extensible validation pipeline (Section 4.1, DG3) provides the architectural hook for domain-specific compliance layers, the logistics compliance agent used in this evaluation is one concrete instantiation of that pattern, described further in Subsection 6.1.1. The theoretical basis for this hypothesis draws on the defence-in-depth principle from information security, the NIST AI Risk Management Framework [8], and W3C SHACL constraint validation [13].

H4 concerns adaptive replanning. *Claim: the framework's multi-tier recovery architecture, comprising bounded retries, LLM-driven strategic replanning, and pheromone-based re-weighting, will achieve reliable failure recovery, escalating to human operators only when automated recovery proves insufficient, and the LLM Strategist will contribute measurable improvements to estimator selection accuracy on assignments.* Its theoretical basis lies in self-healing systems design and bounded-recovery strategies from transaction processing.

Finally, **H5** evaluates the framework as a whole. *Claim: the full framework will outperform a minimal, coordination-free baseline on all quality-related metrics, while incurring measurably higher latency and token cost.* This hypothesis tests whether the additional architectural complexity introduced by the coordination mechanisms is justified by corresponding improvements in measurable system performance.

Hypothesis	Mechanism	Primary Evaluation	Subsection
H1 Stigmergic Learning	Pheromone Engine	Accuracy over successive runs	6.2.2
H2 Competitive Selection	ICNP Bidding	Assignment accuracy drop	6.2.1
H3 Super-additive Safety	Validation Layers	Interaction effect on accuracy	6.2.4
H4 Adaptive Replanning	LLM Strategist	Failure recovery rate	6.2.5
H5 Framework Justification	Full Framework	Accuracy vs. overhead	6.2.1

Table 4: Hypothesis overview. Each hypothesis isolates one coordination mechanism and maps to a dedicated evaluation experiment.

Taken together, these hypotheses align with the design goals defined in Section 4.1. DG1, which concerns decentralised coordination, is examined through H1 on stigmergic learning and H2 on competitive selection. DG3, which addresses safety and governance, is evaluated through H3 on defence-in-depth. DG5, which focuses on adaptive failure recovery, is assessed through H4 on adaptive replanning together with the failure-injection recovery analysis. H5 serves as a holistic and cross-cutting evaluation of the framework under objective O6, testing whether the combined architectural complexity introduced by the coordination mechanisms is justified by measurable improvements over a coordination-free baseline. By contrast, DG2, which provides the ontological substrate, and DG4, which ensures the audit trail, are treated as foundational infrastructure supporting all system configurations and are therefore evaluated implicitly across the quantitative experiments rather than through separate hypotheses.

In the following sections, Section 6.1 outlines the experimental setup, including the domain scenario, ablation configurations, ground truth, and evaluation metrics. Section 6.2 then presents the quantitative evaluation across five experiments. Section 6.3 reports the results of the qualitative user perception survey. Finally, Section 6.4 synthesises the evidence across all five hypotheses and closes the chapter with the consolidated findings.

6.1 Experimental Setup

This section describes the evaluation domain, the agent population instantiated for the experiments, the ablation configurations, the ground-truth definitions, the evaluation metrics, and the execution protocol.

The evaluation is conducted in the commercial warehouse brokerage domain, in which organisations match clients with warehouse facilities based on spatial, regulatory, financial, and temporal constraints. In each pipeline execution, a client submits a logistics request. The system identifies the optimal regional warehouse and deal-estimation strategy, validates the assignment against domain constraints, and produces a client-facing proposal.

Logistics was selected as the evaluation domain for four reasons. First, the domain exhibits high operational complexity, as each brokerage decision must simultaneously satisfy geographic, capacity, compliance, and budgetary constraints. Second, it requires coordination across heterogeneous actors, including regional warehouse specialists, cost analysts, speed-to-operational estimators, and compliance validators, whose expertise is naturally non-overlapping, mirroring the specialised agent populations that Multi-Agent Systems are designed to support. Third, the domain is characterised by frequent exceptions and dynamic re-planning, since client priorities shift, warehouses become unavailable, and regulatory requirements vary by jurisdiction. Fourth, warehouse brokerage has strong practical relevance, as decisions are consequential given lease commitments and

regulatory exposure, auditable given that clients and regulators require justification, and time-sensitive given proposal deadlines and move-in dates.

Although logistics serves as the evaluation domain, the framework is domain-agnostic by design. The coordination mechanisms, namely ICNP, the pheromone engine, SHACL validation, and failure recovery, operate on abstract ontological primitives (`stig:Slot`, `stig:Stigma` and `stig:AffordanceMarker`) that carry no logistics-specific semantics. The domain-specific knowledge resides entirely in the agent implementations and SHACL constraint shapes, both of which are pluggable components. The same architecture is therefore applicable to other domains that share the same structural properties, including healthcare triage, legal document review, financial underwriting, and human resource management and talent acquisition, each of which involves heterogeneous specialised expertise, consequential auditable decisions, and dynamic re-planning under regulatory constraints. The findings therefore constitute evidence about general design principles for robust AI orchestration, not merely results for a single use case.

6.1.1 Domain-Specific Agents

The evaluation pipeline comprises ten domain-specific agents instantiated for the logistics scenario, summarised in Table 5. The agent population was designed to mirror how logistics brokerage is performed in practice, where human experts consult regional warehouse specialists, financial analysts, and timeline analysts depending on the client's binding constraint. Each agent's capabilities and system prompt encode the domain knowledge that a human specialist in that role would apply. The client scenarios, warehouse configurations, and compliance constraints were defined in collaboration with a logistics domain expert. Figure 6 illustrates the pipeline execution DAG.

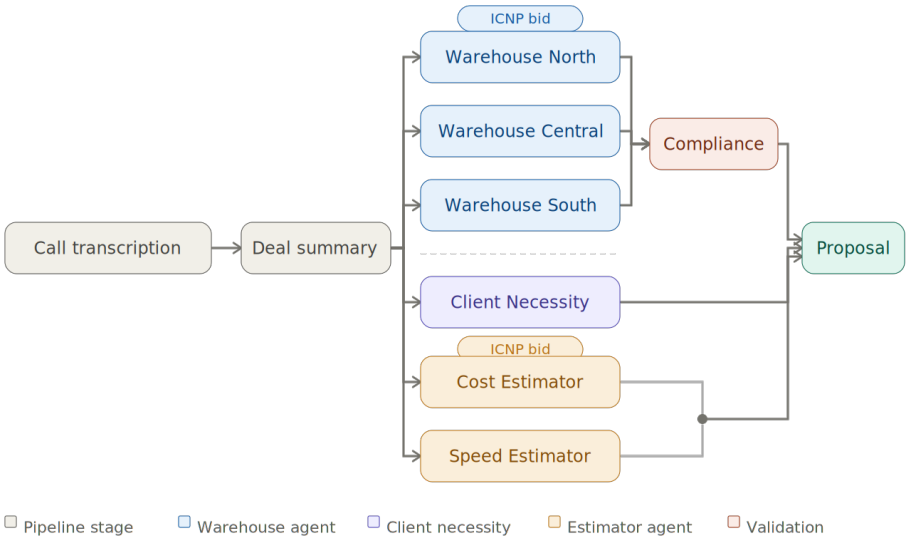


Figure 6: Execution DAG for the warehouse brokerage evaluation pipeline.

All agents follow the same architectural pattern, each implemented as an asynchronous microservice that exposes a uniform entry point, allowing the orchestrator to invoke any agent without knowledge of its internals. Nine of the ten agents are LLM-powered, retrieving domain data from a shared relational database and passing it together with a task-specific system prompt to a large language model. The Compliance Agent is the sole exception, employing deterministic rule logic to guarantee auditability. Each agent

advertises one or more capabilities as string tokens describing what it can do, as listed in Table 5.

Agent	Capabilities	Competes With	LLM	Role
Call Transcription	`transcription`, `call_analysis`	None	✓	Transcribes client calls and extracts structured entities, constraints, and signals that seed the downstream workflow.
Deal Summary	`deal_summary`, `aggregation`	None	✓	Aggregates outputs from upstream agents into a unified deal context that can be reused by subsequent decision agents.
Client Necessity	`form_filling`, `client_analysis`	None	✓	Interprets client requirements and pre-fills requirement forms with operational, spatial, and service needs.
Warehouse North	`warehouse_matching`, `recommendation`, `location_analysis`	Central, South	✓	Evaluates and recommends warehouse options for Northern Europe, specialising in Gothenburg and the broader Scandinavian market.
Warehouse Central	`warehouse_matching`, `recommendation`, `location_analysis`	North, South	✓	Evaluates and recommends warehouse options for the DACH and France corridor, including Munich, Stuttgart, Lyon, and Paris.
Warehouse South	`warehouse_matching`, `recommendation`, `location_analysis`	North, Central	✓	Evaluates and recommends warehouse options for the Italian market, including Milan, Bologna, and Rome.
Cost Estimator	`deal_estimation`, `financial_analysis`	Speed Estimator	✓	Produces financially optimised deal estimates with emphasis on minimising total cost of occupancy.
Speed Estimator	`deal_estimation`, `timeline_analysis`	Cost Estimator	✓	Produces timeline-optimised deal estimates with emphasis on achieving the fastest time to operational outcome.
Compliance	`compliance_check`, `safety_validation`	None	×	Applies deterministic rule-based validation to ensure that recommended options satisfy safety, budget, and operational constraints.
Proposal Template	`proposal_generation`, `document_generation`	None	✓	Assembles validated recommendations and estimates into a structured client-facing proposal document.

Table 5: Agent inventory for the warehouse brokerage evaluation pipeline.

The ten agents are grouped into functional tiers. Table 6 specifies the concrete input and output of each agent.

Agent	Input	Output
Call Transcription	DB: call records (caller, notes, duration, date)	Structured entity extraction: client names, locations, budgets, action items
Deal Summary	DB: deals, emails, notes; upstream: call transcript	Consolidated deal overview: stage, contacts, financial terms, risks, next steps
Client Necessity	DB: client profiles, requirements; upstream: deal summary	Pre-filled necessity form: space, location, budget, special needs (inferred vs. confirmed)
Warehouse North / Central / South	DB: regional warehouse inventory, client requirements; upstream: deal summary	Ranked warehouse shortlist with match scores (0 to 100) and gap analysis
Cost Estimator	DB: available warehouses, client requirements; upstream: deal summary	Cost comparison: monthly rent, lease total, fit-out costs, total cost of occupancy
Speed Estimator	DB: available warehouses, client requirements; upstream: deal summary	Timeline comparison: fit-out durations, compliance lead times, total time to operational
Compliance	DB: client requirements, available warehouses; upstream: suggested pairings (for scoping)	Structured validation report: pass/fail per rule, violation severity (BLOCK/WARN)
Proposal Template	DB: deals, warehouses, client requirements; upstream: all preceding outputs	Client-facing proposal document: executive summary, recommendations, financial terms, timeline

Table 6: Input and output specification per agent. *DB* denotes data retrieved from the shared relational database; *upstream* denotes the output of a preceding agent in the pipeline DAG.

Intake tier. The **Call Transcription** agent retrieves client call records and uses an LLM to extract structured entities such as client names, locations, budgets, and action items. The **Deal Summary** agent aggregates deal records, associated emails, and internal notes into a consolidated deal overview. These two agents execute sequentially at the beginning of the pipeline and produce the shared context consumed by all downstream branches.

Matching tier. Three regional warehouse agents, **North**, **Central**, and **South**, are structurally identical but configured with different geographic filters. Each retrieves only its own regional inventory and matches warehouses against client needs including space, location, budget, and special features, producing a ranked shortlist with match scores. Because all three advertise the same `warehouse_matching` capability, they form a competitive bidding group in which the orchestrator selects the best-scoring agent based on the bid evaluation function described in Section 4.5.

Estimation tier. The **Cost Estimator** and **Speed Estimator** both address deal estimation but optimise for different objectives, namely lowest total cost of occupancy versus fastest time to operational. Because both advertise the `deal_estimation` capability, agent selection follows the same competitive bidding mechanism described in Section 4.5. Each agent encodes domain-specific heuristics in its system prompt, for example fit-out cost formulas for the Cost Estimator and fit-out duration estimates for the Speed Estimator.

Client profiling tier. The **Client Necessity** agent runs in parallel with the matching and estimation tiers, compiling client profiles and requirements into a pre-filled necessity form and flagging inferred versus confirmed fields. Its output feeds directly into the proposal generation stage.

Validation and output tier. The **Compliance** agent validates every client and warehouse pairing against five deterministic business rules covering cold storage, hazmat certification, security level, square footage range, and budget compliance, returning structured pass/fail results. The **Proposal Template** agent performs a fan-in join, consuming the outputs of the compliance check, the client necessity form, and the winning estimator, and generates a client-facing proposal document via LLM.

The agent inventory establishes two competitive bidding points at which the ICNP selection mechanism is exercised. The first is **warehouse matching**, involving a three-way competition among the North, Central, and South agents. The second is **deal estimation**, involving a two-way competition between the Cost and Speed Estimators.

6.1.2 Evaluation Dataset

The evaluation uses a synthetic dataset designed to be fully reproducible and to exercise every coordination mechanism under test. Because no public benchmark exists for Multi-Agent warehouse brokerage, the dataset was constructed manually to satisfy three criteria: (i) each client scenario must have a single unambiguous correct assignment at both competitive bidding points (warehouse selection and estimator selection), enabling deterministic ground-truth evaluation, (ii) the scenarios must span a range of domain constraints (cold storage, hazmat certification, security levels, budget limits) so that the validation layer is exercised, and (iii) the data must include realistic ancillary records (emails, call transcripts, broker notes) to provide sufficient context for the LLM-based agents.

The dataset is seeded into a shared relational database at the start of each experiment and comprises seven tables (Table 7). To ensure that the scenarios reflect realistic brokerage conditions, the dataset was reviewed by a logistics domain expert, who confirmed the plausibility of the client requirements, warehouse configurations, and compliance constraints prior to experimentation.

Table	Records	Content
`clients`	5	Company profile, industry, size, contact details
`client_requirements`	5	Space, budget, location, special needs, priority
`warehouses`	8	Location, capacity, features, rent, security level
`deals`	5	Deal stage, value, assigned broker, notes
`emails`	22	Multi-turn client and broker negotiation threads
`notes`	5	Internal broker observations per deal
`calls`	8	Client and supplier call records with transcripts

Table 7: Evaluation dataset summary. The seven seeded tables provide both the structured records consumed by the LLM-based agents and the realistic ancillary content (emails, calls, broker notes) used by the intake-tier agents.

The five client scenarios span five industries, namely food and beverage, electronics manufacturing, pharmaceuticals, e-commerce fulfilment, and heavy industry, and three European regions covering Southern, Central, and Northern Europe. Each client presents a distinct combination of regional affinity and dominant optimisation objective. FreshCo

Logistics requires cold storage in Milan with a firm budget that favours cost optimisation. TechParts is best served by Central European technical infrastructure in Munich or Stuttgart at moderate cost. GreenLeaf Pharma requires GDP-compliant hazmat storage near Paris or Lyon with a critical pharmaceutical audit deadline that makes time-to-operational the dominant criterion. QuickShip is a cost-driven e-commerce operator suited to Southern European distribution hubs. Nordic Steel AB requires a crane-equipped port facility in Gothenburg, with timeline as the binding constraint due to a critical installation deadline. The eight warehouses listed in Table 8 are distributed across the three regions and vary in available features, creating both clear matches and deliberate mismatches that trigger compliance violations. Three warehouses hold cold-storage certification, namely LogisPark Milano Nord, PharmaStore Lyon, and Paris CDG Logistics, of which two are also hazmat-certified, specifically PharmaStore Lyon and Paris CDG Logistics.

Warehouse	City (Region)	Avail. sqft	Cold	Hazmat	Security	€/sqft
LogisPark Milano Nord	Milan (South)	22,000	✓	—	enhanced	0.75
Centro Logistico Bologna	Bologna (South)	55,000	—	—	standard	0.65
Roma Sud Distribution	Rome (South)	48,000	—	—	standard	0.60
Bavaria Logistics Hub	Munich (Central)	40,000	—	—	standard	0.85
Stuttgart TechCenter	Stuttgart (Central)	35,000	—	—	high	0.90
PharmaStore Lyon	Lyon (Central)	28,000	✓	✓	high	0.95
Paris CDG Logistics	Roissy (Central)	25,000	✓	✓	high	1.10
Gothenburg Port Warehouse	Gothenburg (North)	60,000	—	—	enhanced	0.70

Table 8: Warehouse inventory used in the evaluation dataset. Cold and Hazmat flags denote certified cold-chain and hazardous-materials storage, respectively. Security levels are `standard`, `enhanced`, and `high`. Rent is the monthly rate per available square foot (EUR).

The email and call records simulate realistic brokerage interactions, including requirement discovery, specification exchanges, pricing negotiations, and compliance discussions. These records serve as the input context for the intake-tier agents (Call Transcription, Deal Summary) and provide a realistic information environment without introducing the noise of real-world data that would confound the quantitative analysis.

Ground-truth assignments were derived from the seed data. Each client has a unique regional affinity and a dominant optimisation objective (cost versus speed) that follows directly from the requirements introduced in Subsection 6.1.2. Table 9 defines the expected winner for each client at both competitive bidding points.

Client	Industry	Priority	Correct Warehouse	Correct Estimator
FreshCo	Food distribution	High	Warehouse South	Cost Estimator
TechParts	Electronics	High	Warehouse Central	Cost Estimator
GreenLeaf	Pharmaceuticals	Critical	Warehouse Central	Speed Estimator
QuickShip	E-commerce	Medium	Warehouse South	Cost Estimator
NordicSteel	Heavy industry	High	Warehouse North	Speed Estimator

Table 9: Ground-truth agent assignments for the five evaluation scenarios, used as the reference against which assignment accuracy (ACC) is measured throughout the ablation study. Priority levels are ordered Critical, High, Medium.

Two of the five clients are seeded with warehouse assignments that deterministically trigger domain-rule violations, independent of configuration or coordination mechanism.

Table 10 defines the ground-truth violation status per client. This yields a ground-truth violation rate of $2/5 = 40\,\%$.

Client	Priority	Rule	Injected Assignment	Severity
FreshCo	High	`COLD_STORAGE`	Bavaria Hub	`BLOCK`
FreshCo	High	`COLD_STORAGE`	Bologna	`BLOCK`
GreenLeaf	Critical	`HAZMAT`	Bologna	`BLOCK`
GreenLeaf	Critical	`HAZMAT`	LogisPark Milano	`BLOCK`
GreenLeaf	Critical	`HIGH_SECURITY`	Bologna	`BLOCK`
GreenLeaf	Critical	`HIGH_SECURITY`	Bavaria Hub	`BLOCK`
TechParts	High	—	—	Pass
QuickShip	Medium	—	—	Pass
NordicSteel	High	—	—	Pass

Table 10: Injected client and warehouse pairings used to exercise each blocking rule, yielding a ground-truth violation rate of 40%. FreshCo and GreenLeaf pairings are seeded to guarantee a deterministic violation, while TechParts, QuickShip, and NordicSteel have no seeded violation. Warehouses use the short forms defined in Table 8, and priority levels are ordered Critical, High, Medium.

In addition to the violations observed during the main ablation runs, a separate targeted validation probe verifies that the SHACL and Compliance layers detect non-overlapping defect classes. The probe injects ten known cases (see Table 11) into plan drafts, comprising six structural defects, three domain defects, and one negative control representing a valid pairing, and checks which layer catches each one. Unlike the main experiment, the probe runs deterministically outside the pipeline with no LLM calls and no CNP bidding.

ID	Defect Injected	Class	Expected Layer
S1	Missing `assigned_agent` field	Structural	SHACL
S2	Empty capabilities list	Structural	SHACL
S3	Missing task description	Structural	SHACL
S4	Hazmat task without human approval	Structural	SHACL
S5	Cold-storage task without temperature range	Structural	SHACL
S6	Critical deal without high-risk flag	Structural	SHACL
D1	FreshCo → Bavaria Hub (no cold storage)	Domain	Compliance
D2	GreenLeaf → Bologna (no hazmat, std. security)	Domain	Compliance
D3	GreenLeaf → LogisPark Milano (no hazmat)	Domain	Compliance
D4	QuickShip → Lyon (valid pairing)	Control	None

Table 11: Validation probe ground truth, comprising six structural defects expected to be caught by SHACL, three domain defects expected to be caught by the Compliance Agent, and one valid pairing as a negative control. Warehouses use the short forms defined in Table 8.

Of the ten cases, six are structural defects that only SHACL can detect, three are domain violations that only the Compliance Agent can detect (D1–D3), and one is a clean pairing included as a negative control (D4).

6.1.3 Metrics

The experiments reported in this chapter rely on three sets of metrics, introduced in turn across the ablation study, the deep-dive analyses, and the qualitative user perception survey.

The five primary metrics used in the ablation study are defined in Table 12.

Metric	ID	Type	Formula / Source
Assignment Accuracy	ACC	Ratio $[0, 1]$	Correct assignments divided by total assignments, computed from `ContractAward.bidder_id` against ground truth
Constraint Violation Rate	VR	Ratio $[0, 1]$	Runs with at least one `BLOCK` violation divided by total runs, measured from plan validator and compliance output
Recovery Success Rate	RSR	Count (n/N)	Recovered pipelines divided by total failure injections
End to End Latency	LAT	Continuous (s)	Time from pipeline start to audit log timestamp
Token Consumption	TOK	Count	Sum of LLM input and output tokens per-run, taken from OpenAI API usage

Table 12: Primary metrics used in the ablation study. ACC and VR measure assignment quality and safety respectively, RSR measures recovery effectiveness, and LAT and TOK measure deployment cost.

Secondary metrics, used in the deep-dive analyses (Sections 6.2.2–6.2.5) to provide fine-grained evidence for each hypothesis, are defined in Table 13.

Metric	ID	Type	Description
Convergence Run	CONV	Count	Run number at which ACC exceeds 0.85 and remains above for five subsequent runs
Pheromone Intensity	PHI	Continuous $[0, \infty)$	`AffordanceMarker.intensity` per agent per task type; measures learned preference strength
Bid Score Delta	BSD	Continuous $[0, 1]$	Winning bid score minus runner up; measures the competition margin between agents
Interaction Effect	IE	Continuous	ACC($C0$) − ACC($C3$) − ACC($C4$) + ACC($C3$+$C4$); tests whether validation layers interact super-additively on downstream accuracy
True Positive Rate	TPR	Ratio $[0, 1]$	Correctly detected violations divided by total ground truth violations
False Positive Rate	FPR	Ratio $[0, 1]$	Incorrectly flagged clean assignments divided by total clean assignments
Recovery Time	RT	Continuous (s)	Elapsed time from injected failure to pipeline completion, with a 300 s timeout
Estimator Win Ratio	EWR	Ratio	Cost Estimator wins divided by total estimator selections, stratified by client priority level

Table 13: Secondary metrics used in the deep-dive analyses, providing fine-grained evidence for each hypothesis beyond the primary ablation metrics.

The qualitative metrics used in the user perception survey (Section 6.3), appropriate for ordinal Likert-scale data, are summarised in Table 14.

Metric	Type	Description
Likert rating	Ordinal $[1, 5]$	5 point scale ($1 =$ Strongly Disagree … $5 =$ Strongly Agree) applied to 19 survey items across output quality, decision transparency, framework value, and overall usefulness
Forced choice preference	Categorical	Participants select the most trusted output and the best audit trail among three anonymised system configurations
Comparative ranking	Ordinal $[1, 3]$	Full ranking of three outputs from best (1) to worst (3)
Thematic frequency	Count (k/N)	Number of participants whose open ended responses contain a given thematic code, retained if $k \geq 3$
Descriptive statistics	Summary	Median (Mdn) as primary measure of central tendency, with mean (M) and standard deviation (SD) reported for comparability and interquartile range (IQR) or full frequency distributions reported where ceiling or bimodal patterns would otherwise be obscured

Table 14: Metrics used in the qualitative user perception survey, designed for ordinal Likert data and small-sample comparative analysis (N = 19).

6.1.4 Statistical Approach

Non-parametric methods are used throughout, consistent with the ordinal and non-normally distributed nature of the data. The complete set of tests, their purpose, and the corresponding effect size measures are summarised in Table 15. The domain-versus-technical contrast in the qualitative survey is treated as an a priori comparison without multiple-comparisons correction, with findings interpreted cautiously given the modest sample of $N = 19$.

Test	Purpose	Effect Size and Thresholds
Wilcoxon signed-rank [92]	Pairwise comparison of full framework against each ablation ($N = 5$ paired client means), with Bonferroni correction [93] ($\alpha_{\text{Bonf}} = 0.0063$)	Underpowered at $N = 5$, reported alongside Cliff's δ
Cliff's δ [94]	Power-independent effect size over all 50 runs per configuration pair	< 0.147 negligible, 0.147–0.33 small, 0.33–0.474 medium, $\geq$ 0.474 large
Chi-square [95]	Association between client priority and selected estimator	Cramér's V [96]
Friedman [97]	Differences across C0, C3, C4, and C3+C4	Kendall's W
Mann–Whitney U [98]	Domain experts versus technical evaluators in the qualitative survey	$r = \lvert Z \rvert / \sqrt{N}$ [99], with ≥ 0.1 small, ≥ 0.3 medium, ≥ 0.5 large

Table 15: Statistical tests used across the evaluation. Significance level $\alpha = 0.05$ throughout.

6.1.5 Execution Protocol and Configurations

The study comprises 790 total pipeline executions, distributed across three experimental phases. The ablation study (Subsection 6.2.1) accounts for 450 runs (9 configurations × 5 clients × 10 runs). The pheromone convergence experiment (Subsection 6.2.2) contributes 300 runs (2 configurations × 5 clients × 30 runs). The failure injection experiment (Subsection 6.2.5) contributes 40 runs (4 configurations × 10 injections), with recovery timeout set to 300 s.

All LLM-based agents in the evaluation use the OpenAI GPT-4o-mini model [4] (`gpt-4o-mini`). GPT-4o-mini is a small-footprint member of the GPT-4o family, specifically an instruction-tuned, transformer-based chat model with a 128 K-token context window and a maximum of 16 384 output tokens per call. It provides a favourable balance between instruction-following quality and per-token cost, making it appropriate for a large-scale ablation study where per-run token expenditure is a controlled variable (metric TOK). The model was selected because it reliably produces structured outputs, as required by the ICNP bid format and the SHACL-validated plan schema, without additional fine-tuning. Table 16 summarises the full inference configuration applied uniformly across all agents and all configurations.

Parameter	Value
Provider	OpenAI
Model ID	`gpt-4o-mini`
Model family	GPT-4o (instruction-tuned, chat)
Context window	128 000 tokens
Max output tokens	16 384 tokens per call
Temperature	0
API version / SDK	`openai` $\geq$ 1.59 (`Chat.completions.create`)
Concurrency limit	10 simultaneous requests (semaphore-controlled)
Timeout per call	30 s

Table 16: LLM inference configuration applied uniformly across all agents and all ablation configurations. Temperature is set to 0 to minimise stochastic variance, with residual non-determinism addressed through 10-run replication per scenario.

Temperature is set to 0 for all experiments, minimising stochastic variance in agent outputs while retaining realistic execution characteristics such as structured reasoning and multi-step inference. Residual non-determinism from batched inference scheduling and floating-point arithmetic at the provider side is addressed by the 10-run replication per scenario per configuration.

Regarding pheromone persistence, in the ablation study each configuration runs 10 consecutive executions per client. For the full framework, pheromone markers persist across all 10 runs, enabling the engine to accumulate reinforcement signal. For the pheromone-disabled configuration (C1), markers are cleared between every run to simulate a stateless system. For all configurations where pheromone state is not the independent variable under test (C2–C7), markers are cleared between configurations but persist across the 10 runs within each configuration, matching the default C0 behaviour. In the convergence experiment, the full-framework markers persist across all 30 runs per client. Markers in the pheromone-disabled condition are cleared between every run to simulate a stateless system. Configuration labels are defined in Subsection 6.2.1.

6.2 Quantitative Analysis

The quantitative analysis is organised into four experiments, each designed to isolate a distinct coordination mechanism and test one or more of the hypotheses stated in the beginning of Chapter 6. Subsection 6.2.1 presents the ablation study, which systematically removes one framework component at a time across nine configurations to quantify each mechanism's marginal contribution to assignment accuracy, violation detection, latency, and token consumption, addressing H2 and H5. Subsection 6.2.2 examines pheromone convergence through a 30-run longitudinal experiment comparing persistent and cleared pheromone state, testing whether stigmergic learning produces the characteristic accuracy trajectory predicted by H1. Subsection 6.2.3 analyses priority-based agent selection using C0 data alone, examining whether the ICNP's competitive bidding mechanism develops context-sensitive routing behaviour as an emergent property rather than an explicitly programmed rule. Subsection 6.2.4 investigates the interaction between the SHACL structural validator and the Compliance Agent through a 2×2 factorial design and a targeted defect-injection probe, testing the super-additive defence-in-depth hypothesis H3. Subsection 6.2.5 completes the analysis with a failure-injection study across four configurations, evaluating the multi-tier recovery mechanism and testing H4 on adaptive replanning under controlled failure conditions.

6.2.1 Ablation Study

The ablation study constitutes the central quantitative contribution of this evaluation. By systematically removing one framework component at a time, it provides direct empirical evidence for the marginal contribution of each coordination mechanism to the system's overall effectiveness, isolating what each component adds rather than measuring the system as an opaque whole. Nine configurations are evaluated, each removing exactly one framework component from the full system while holding all other factors constant.

6.2.1.1 Configurations and Aggregate Results

The nine ablation configurations are summarised in Table 17. C1 isolates the pheromone engine's contribution by preserving all other components, so that the only difference is whether assignment history modulates future bid scores. C2 replaces ICNP bidding with round-robin allocation, the simplest non-random static policy. C3 and C4 are analysed jointly (as C3+C4) and separately to distinguish structural from domain validation contributions, with C4 removing a domain-specific rule-based service that belongs to the logistics scenario rather than to the orchestration framework. C7 provides a lower bound by stripping all coordination to expose the maximum possible degradation attributable to the framework. Appendix A provides a step-by-step trace of a representative FreshCo run under C0 to ground the aggregate results in a concrete execution.

Table 18 presents the primary metrics across all nine configurations, with $N = 50$ runs per configuration (5 clients $\times$ 10 runs). Several notable patterns emerge from these results. The IQR column carries more information than a simple dispersion measure. The modal IQR for high-performing configurations (C0, C1, C3+C4, C5) is 1.00–1.00, meaning the lower and upper quartiles coincide at the maximum. The accuracy distribution is left-skewed, with most runs achieving perfect assignment and a minority producing partial or zero-accuracy outcomes. C2's IQR of 0.00–0.50 shows the opposite pattern. At least half of C2 runs produce zero correct assignments, and the distribution is concentrated at the lower end. C7's IQR of 0.00–1.00 reveals bimodality, since some scenarios are

solved perfectly even without coordination (because the LLM happens to generate the right output), while others fail completely. This pattern is consistent with the hypothesis that the framework's mechanisms suppress variance as much as they increase the mean.

Config	Component Removed	Implementation
C0	None (full framework)	Default configuration, all components enabled.
C1	Pheromone adaptation	Pheromone engine skipped in pipeline DAG, markers cleared between runs.
C2	ICNP bidding	Round-robin assignment, no ICNP scoring.
C3	SHACL validation	SHACL ruleset path set to empty, plan validator skips structural checks.
C4	Compliance agent	Domain-specific rule-based service (part of the logistics scenario, not the orchestration framework) removed from pipeline DAG.
C3+C4	SHACL + Compliance	Both C3 and C4 changes applied simultaneously.
C5	LLM Strategist	LLM Strategist skipped in pipeline DAG, no runtime graph adaptations.
C6	Ethics Checker	Ethics-check service skipped in pipeline DAG.
C7	All coordination	Direct sequential agent calls, no ICNP, no pheromone markers, no validation, no audit logging.

Table 17: Ablation configuration summary.

Config	ACC Mean (IQR)	WH ACC	EST ACC	VR	LAT Mdn (IQR)	TOK Mdn
C0	**0.98** (1.00–1.00)	**100%**	96%	**40%**	210 (177–239)	33,572
C1	**0.98** (1.00–1.00)	**100%**	96%	**40%**	192 (166–214)	31,888
C2	0.42 (0.00–0.50)	34%	50%	44%	**154** (146–177)	32,752
C3	0.85 (0.50–1.00)	88%	82%	**40%**	174 (162–183)	31,044
C4	0.90 (1.00–1.00)	88%	92%	0%	167 (149–189)	**30,156**
C3+C4	**0.98** (1.00–1.00)	96%	**100%**	0%	187 (172–195)	31,116
C5	0.95 (1.00–1.00)	**100%**	90%	**40%**	162 (153–173)	31,851
C6	0.91 (1.00–1.00)	94%	88%	**40%**	166 (152–182)	31,178
C7	0.50 (0.00–1.00)	40%	60%	0%	197 (184–210)	30,599

Table 18: Ablation study component contribution summary. Metric definitions are given in Table 12. Mean is the arithmetic mean of run-level accuracy, IQR is the interquartile range over all 50 individual runs, not over the per-client means.

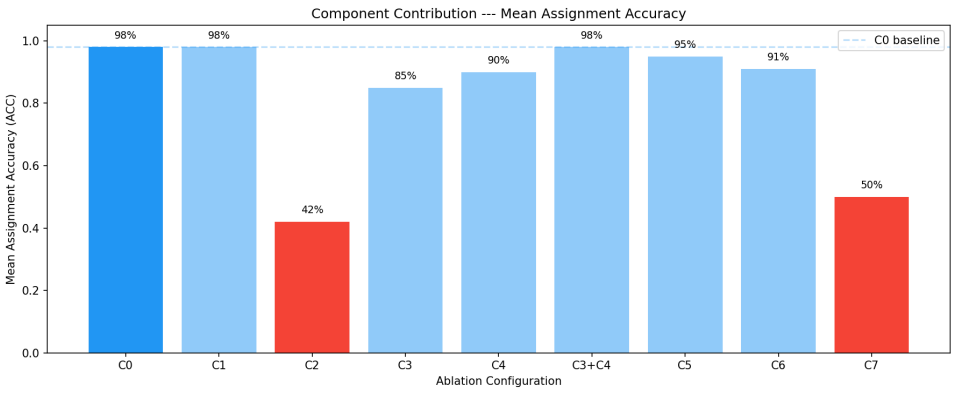


Figure 7: Mean assignment accuracy by ablation configuration.

6.2.1.2 Coordination Effects

Under round-robin allocation (Table 18, row C2), warehouse accuracy (34%) and estimator accuracy (50%) correspond almost exactly to the theoretical random baselines for 3-way selection ($1/3 \approx 33\%$) and 2-way selection ($1/2 = 50\%$), respectively. The 1-point surplus over chance in WH accuracy and zero surplus in EST accuracy confirm that round-robin adds no information beyond random selection. This finding falsifies the alternative explanation that individual agent LLM quality, rather than the coordination mechanism, drives C0's accuracy. If agents were intrinsically superior, even round-robin allocation would produce above-chance accuracy.

C2 is also the only configuration with a violation rate above 40%, a 4 pp surplus driven entirely by TechParts and dissected in Subsection 6.2.4.

The minimal baseline C7 achieves 0.50 overall, comprising WH = 40% and EST = 60%. The warehouse accuracy (40%) slightly exceeds the random baseline (33%), while the estimator accuracy (60%) exceeds chance (50%) by 10 pp on 10 runs, a surplus whose magnitude should be read cautiously given the small sample. This surplus arises because C7 still relies on GPT-4o-mini's internal reasoning, meaning that even without the ICNP mechanism the LLM occasionally selects the contextually appropriate agent when given direct task instructions. The EST surplus (10% above chance) is larger than the WH surplus (7%), consistent with the estimator decision being simpler, since cost versus speed maps directly to budget versus timeline signals in the task description, while warehouse routing requires geographical and compliance knowledge that the LLM does not reliably internalise without pheromone guidance. Figure 8 visualises the per-configuration accuracy gap relative to C0, showing that C2 and C7 stand apart from the cluster of high-performing configurations.

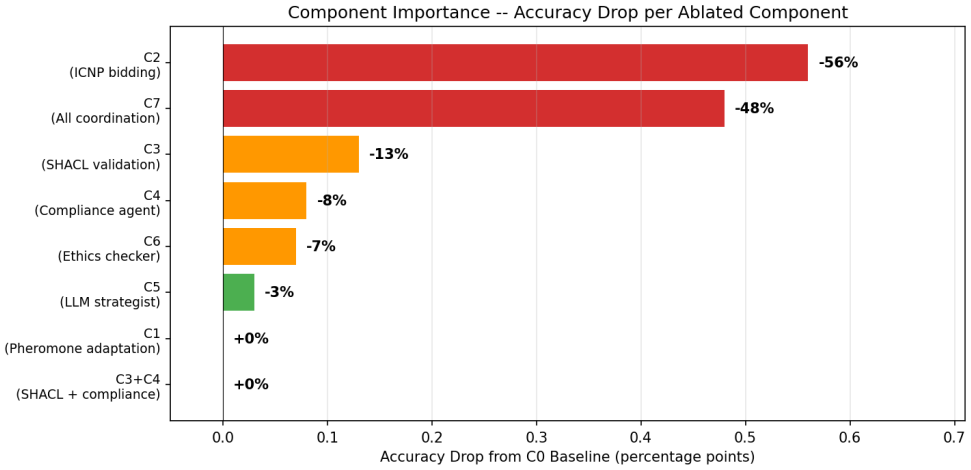


Figure 8: Accuracy delta (C0 − C_x) by configuration.

Client	Runs	WH Winner	WH Correct	EST Winner	EST Correct
FreshCo	90	warehouse-south	76/90	cost-estimator	82/90
TechParts	90	warehouse-central	73/90	cost-estimator	76/90
GreenLeaf	90	warehouse-central	76/90	speed-estimator	78/90
QuickShip	90	warehouse-south	78/90	cost-estimator	84/90
NordicSteel	90	warehouse-north	**67/90**	speed-estimator	**57/90**

Table 19: Per-client assignment accuracy across all 9 ablation configurations (90 runs per client). NordicSteel emerges as the most coordination-sensitive scenario. Ground-truth assignments are defined in Table 9.

Table 19 reports per-client assignment accuracy aggregated across all ablation configurations. Accuracy is relatively stable across FreshCo, TechParts, GreenLeaf, and QuickShip, with each individual decision metric for these four clients falling between 73 and 84 correct out of 90 runs, compared with NordicSteel's lowest value of 57.

The notable outlier is NordicSteel, which produces the lowest per-client accuracy across both decisions, specifically 67/90 warehouse selections and 57/90 estimator selections. The warehouse routing is relatively straightforward (Warehouse North, Gothenburg), but the estimator selection is counter-intuitive from the LLM's perspective, since a steel client with heavy industrial machinery intuitively maps to cost optimisation while the ground truth requires the Speed Estimator due to a critical crane installation deadline. NordicSteel is also the slowest client to converge in the pheromone analysis (Subsection 6.2.2), making it the scenario where learned pheromone guidance most clearly adds value over uninformed LLM reasoning.

The NordicSteel column in Figure 9 confirms this sensitivity. Unlike GreenLeaf, whose failures concentrate in C7 alone, NordicSteel shows consistently degraded performance across mid-tier configurations (C3, C4, C6), identifying it as the scenario most sensitive to which coordination mechanisms are active.

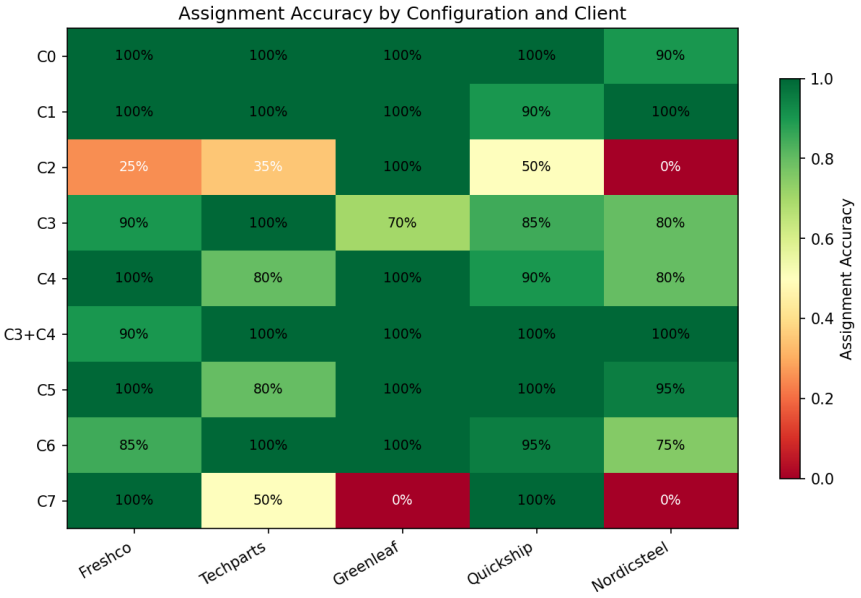


Figure 9: Assignment accuracy heatmap with configurations as rows and clients as columns.

Removing both validation layers (C3+C4) leaves ACC unchanged at 0.98, yet VR drops from 40% to 0%. This apparent contradiction is explained in Subsection 6.2.4, which analyses the validation layer interaction in detail and shows that assignment correctness (ACC) and constraint compliance (VR) are orthogonal quality dimensions evaluated by different agents at different pipeline stages. The ICNP selects the correct warehouse regardless of whether a downstream validation layer exists, so the pipeline can be accurate but unsafe. The joint metric pair (ACC, VR) is therefore not redundant, as each captures a failure mode that the other cannot detect.

6.2.1.3 Cost Profile

The Pareto frontier (Figure 10) reveals that the two configurations suffering the largest accuracy penalties, C2 and C7, do not recover that cost in latency savings. C7 combines low accuracy (0.50) with high latency (197 s), occupying the worst position on both dimensions. C5 represents the most efficient configuration, achieving 0.95 accuracy at the lowest

latency among high-performing configurations (162 s). Table 20 ranks configurations by median end-to-end latency.

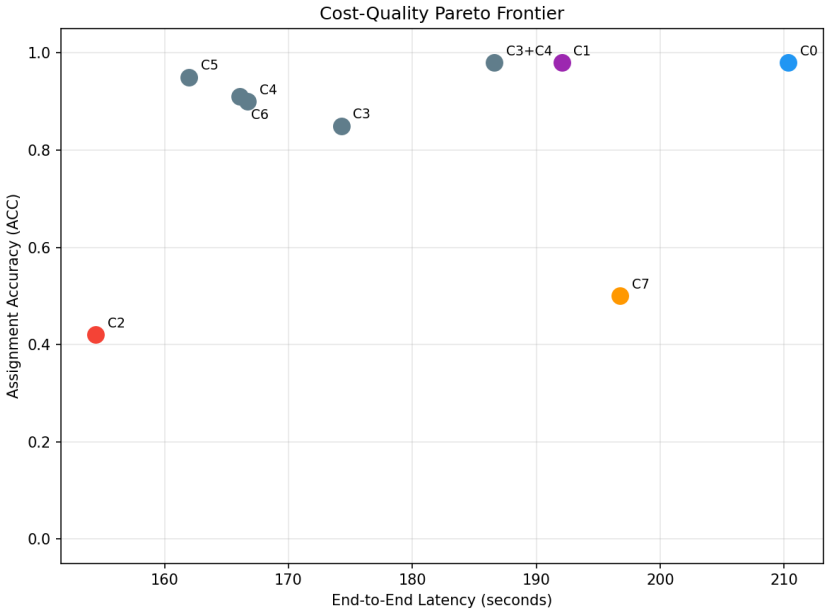


Figure 10: Cost-quality Pareto frontier showing accuracy against end-to-end latency across ablation configurations. C5 occupies the most efficient position at 0.95 accuracy and 162 s, while C7 is dominated on both dimensions.

Config	Median (s)	IQR (s)	Δ vs. C0	Primary saving
C2 (−ICNP)	**154**	**146–177**	**−56 s (−27%)**	**No bidding rounds**
C5 (−strategist)	162	153–173	−48 s (−23%)	No strategy LLM call
C6 (−ethics)	166	152–182	−44 s (−21%)	No ethics LLM call
C4 (−compliance)	167	149–189	−43 s (−21%)	No compliance scan
C3 (−SHACL)	174	162–183	−36 s (−17%)	No SHACL validation
C3+C4 (−both)	187	172–195	−23 s (−11%)	No validation layers
C1 (−pheromone)	192	166–214	−18 s (−9%)	No pheromone I/O
C7 (minimal)	197	184–210	−13 s (−6%)	No coordination
C0 (full)	210	177–239	reference	Baseline

Table 20: End-to-end latency by configuration, ranked fastest to slowest. The full framework C0 carries a 6.6% latency overhead over the minimal baseline C7, with ICNP bidding (C2) and the LLM Strategist (C5) identified as the two largest single contributors.

Three patterns emerge from the latency ranking. First, ICNP bidding is the single largest latency contributor, as removing ICNP (C2) saves 56 s, more than any other single ablation. Although the bid scoring itself is purely numerical (a weighted combination of trust, cost, and capability match, computed by the centralised blueprint bidder), the ICNP incurs latency through marker-store queries for pheromone markers, historical success rates, and failure penalties for each candidate agent, followed by a scoring aggregation and ranking step. Second, the LLM Strategist and Ethics Checker each contribute approximately 45 s of latency through their respective LLM inference calls, yet both produce negligible effects on assignment accuracy. This identifies both components as candidates for optimisation, as they impose comparable latency costs while contributing little to assignment quality. Third, C7 is not the fastest configuration despite removing

all coordination. It is in fact slower than every other ablation configuration because it runs agents sequentially without the ICNP's ability to parallelise independent pipeline branches. The coordination overhead present in C0 is therefore partially offset by the orchestration benefits it provides, making full coordination removal a poor strategy for latency reduction.

The IQR column also reveals a distributional pattern, with C0 exhibiting the widest variability (IQR = 62 s), while C5 has the tightest (IQR = 20 s). Since C5 retains ICNP bidding but removes the LLM Strategist, this suggests that the primary source of latency variance is the Strategist's non-deterministic LLM inference calls rather than the ICNP's numerical bid scoring. This is further confirmed by C1 (IQR = 48 s, Strategist enabled) compared with C2 (IQR = 31 s, Strategist enabled but no ICNP), as removing ICNP reduces variance moderately, but removing the Strategist (C5) reduces it the most.

Figure 11 confirms the distributional patterns reported in Table 20. C0 exhibits the widest spread, with several high-latency outliers exceeding 400 s that pull the distribution upward while leaving the median at 210 s, indicating that extreme latency events are infrequent but non-negligible in the full framework. The cluster of configurations C3, C4, C5, and C6 visually occupies a tight band between 162–174 s with narrow boxes and few outliers, confirming that removing any one of these components produces comparable and predictable latency savings without introducing additional variance.

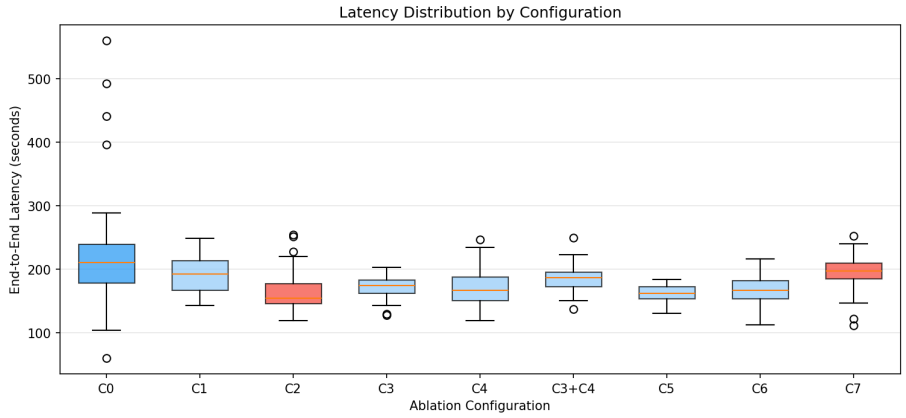


Figure 11: End-to-end latency distribution by configuration. C0 exhibits the widest spread with several outliers above 400 s, while configurations C3 to C6 cluster tightly between 162 and 174 s, indicating that single-component removals produce predictable latency savings without added variance.

Table 21 and Figure 12 rank configurations by median token consumption and reveal a pattern that differs markedly from the latency ranking. The Compliance Agent is the most token-expensive component to remove (−10%), yet it contributes zero LLM tokens itself since it is deterministic. The explanation is that agents incorporate compliance context into their upstream reasoning when the Compliance Agent is active, inflating prompt length across the pipeline. When the agent is absent, this contextual overhead disappears, producing the largest aggregate token saving of any single component removal.

A notable feature of the ranking is that C4 consumes fewer tokens than C7, the minimal baseline, despite C7 removing all coordination mechanisms and C4 removing only the Compliance Agent. This reflects the nature of C7's sequential execution. Without the structured constraints imposed by ICNP bidding, agents in C7 produce unguided, verbose completions rather than bounded bid responses, inflating token consumption even in the absence of coordination overhead. The compliance removal effect in C4 therefore operates on top of a pipeline that is already more token-disciplined than the coordination-free baseline, which explains why a partially dismantled framework can outperform a fully

stripped one on this metric.

Config	Median tokens	Δ vs. C0	Primary saving
C4 (− compliance)	30,156	−3,416 (−10%)	No compliance context in upstream prompts
C7 (minimal)	30,599	−2,973 (−9%)	No coordination
C3 (− SHACL)	31,044	−2,528 (−8%)	No SHACL constraint context in plan generation
C3+C4 (− both)	31,116	−2,456 (−7%)	No validation layers
C6 (− ethics)	31,178	−2,394 (−7%)	No ethics prompt
C5 (− strategist)	31,851	−1,721 (−5%)	No strategy prompt
C1 (− pheromone)	31,888	−1,684 (−5%)	No pheromone context
C2 (− ICNP)	32,752	−820 (−2%)	No bid scoring
C0 (full)	33,572	Reference	Baseline

Table 21: Median token consumption by configuration, ranked from most efficient to least efficient. The Compliance Agent has the largest token footprint despite contributing zero LLM tokens itself, since its presence inflates compliance context across upstream prompts.

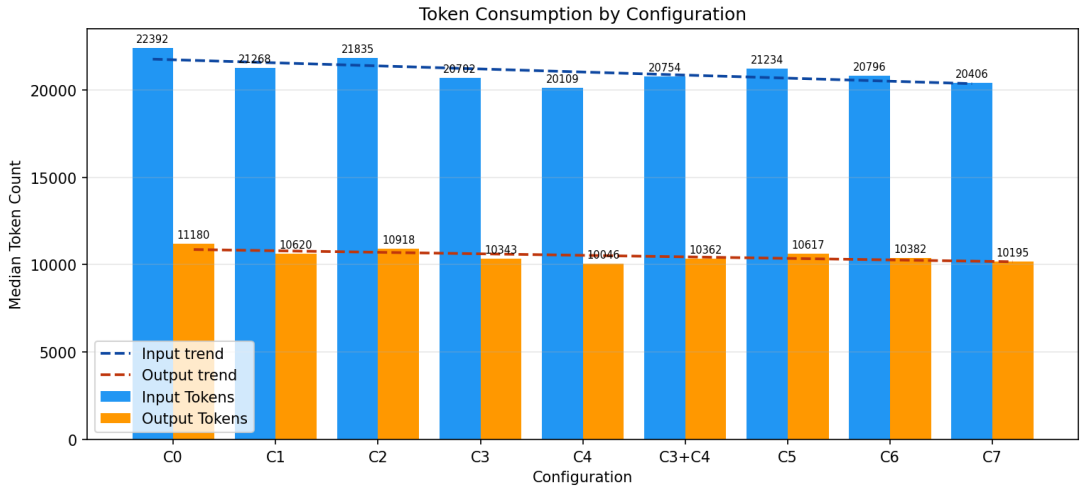


Figure 12: Median input and output token consumption per configuration. The Compliance Agent has the largest token footprint despite contributing zero LLM tokens itself, since its presence inflates compliance context across upstream prompts.

Conversely, ICNP bidding, the largest latency contributor, is the smallest token contributor (−2%), because bid scoring relies on lightweight numerical computation rather than LLM inference. This asymmetry has direct deployment implications. If latency is the binding constraint, disabling ICNP yields the largest speedup but sacrifices 56 percentage points of accuracy. If token cost is the binding constraint, disabling the Compliance Agent yields the largest token saving but eliminates violation detection entirely. Neither trade-off is acceptable in isolation, which reinforces the conclusion that the full framework's overhead is justified by its joint quality and safety guarantee.

The total token overhead of the full framework over the minimal baseline is bounded. C0 consumes 33,572 median tokens against 30,599 for C7, a difference of 2,973 tokens (+9.7%). At current API pricing this overhead is modest relative to the baseline cost of a single pipeline run. The primary drivers of token consumption are the number of competing agents and bidding rounds, meaning that cost growth is determined by coordination breadth rather than by pipeline depth or data volume. This makes the framework's token overhead predictable as the agent population grows, although empirical confirmation across variable agent populations is reserved for future work.

6.2.1.4 Statistical Significance and Overall Validation

Wilcoxon signed-rank tests comparing C0 against each ablation configuration yield uniformly non-significant p-values, with the raw statistics reported in Appendix C. This reflects a statistical power limitation rather than an absence of meaningful effects.

Each Wilcoxon comparison pairs the *per-client mean accuracy* of C0 against that of an ablation configuration C_x, yielding $N = 5$ paired observations per test (one per evaluation client). Because accuracy is sharply skewed toward the ceiling, with most configurations scoring at or above 85% on three to four of the five clients, the ranks compress and the test cannot achieve the power needed to detect meaningful differences at $\alpha = 0.05$. This is a sample-size limitation intrinsic to the five-client evaluation design, not evidence of equivalence between configurations. Table 22 therefore reports Cliff's delta computed over all $N = 50$ individual runs per configuration, providing a power-independent measure of practical significance at the run level rather than the client level. Because every raw Wilcoxon p-value already sits above the uncorrected $\alpha = 0.05$ threshold (Appendix C), the choice of multiplicity correction does not alter the qualitative conclusion. The limitation is one of statistical power rather than family-wise error, and future work should expand the evaluation to at least 15–20 client scenarios to achieve adequate power at the client level.

Config	Component	C0 ACC	C_x ACC	Δ (C0 $-$ C_x, pp)	Cliff's δ	
C1	Pheromone adaptation	0.98	0.98	+0	0.000	(negligible)
C2	ICNP bidding	0.98	0.42	+56	**0.774**	(large)
C3	SHACL validation	0.98	0.85	+13	0.260	(small)
C4	Compliance agent	0.98	0.90	+8	0.160	(small)
C3+C4	Both validation	0.98	0.98	+0	0.000	(negligible)
C5	LLM Strategist	0.98	0.95	+3	0.060	(negligible)
C6	Ethics Checker	0.98	0.91	+7	0.121	(negligible)
C7	All coordination	0.98	0.50	+48	**0.576**	(large)

Table 22: Cliff's delta effect sizes C0-against-C_x comparisons ($N = 50$ runs per configuration).

Two large effects are confirmed. ICNP bidding ($\delta = 0.774$) and full coordination removal ($\delta = 0.576$) both reach the large threshold. These results establish the ICNP as the framework's most impactful single mechanism and the full coordination stack as providing substantial, practically significant improvements over uncoordinated execution. The negligible effect size for C3+C4 ($\delta = 0.000$) is explained by a super-additive interaction analysed in Subsection 6.2.4. This pattern is invisible to pairwise C0-against-C_x comparisons.

The C0 against C7 comparison (Figure 13) provides the clearest test of whether the framework's architectural complexity is warranted. Across all five metrics, C0 dominates C7 decisively. Assignment accuracy rises from 50% to 98%, warehouse correctness from 40% to 100%, estimator correctness from 60% to 96%, compliance detection from 0% to 40%, and plan validity from 0% to 94%. These gains are achieved at a cost of only 6.6% additional latency and 9.7% additional token consumption. The plan validity result is particularly noteworthy. C7's complete absence of SHACL validation and governance mechanisms means that every plan it generates is structurally unverified, a condition that would be unacceptable in any regulated deployment context. The token overhead reflects the additional prompt context injected by the coordination layer, including pheromone state, bid summaries, compliance reports, and pheromone marker writes. At the scale of individual pipeline runs these are negligible costs relative to the baseline token budget of a

30,000-token pipeline, and coordination overhead is bounded by the number of competing agents and the depth of the bidding protocol rather than by the pipeline's total complexity. This evidence directly challenges the common objection that Multi-Agent coordination incurs prohibitive overhead relative to its quality contribution.

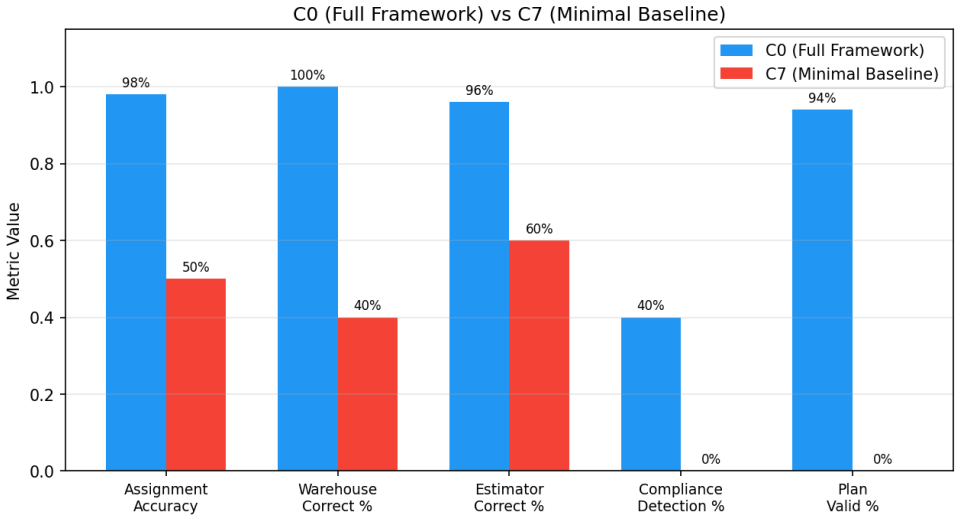


Figure 13: C0 against C7 side-by-side comparison across all primary metrics. The full framework dominates the coordination-free baseline on every quality dimension at a cost of only 6.6% additional latency and 9.7% additional token consumption.

6.2.2 Pheromone Convergence

To test **H1** (stigmergic learning), C0 and C1 each execute 30 consecutive runs per client with persistent (C0) or cleared (C1) pheromone state. Table 23 reports the per-client convergence points and final accuracy for both configurations.

Client	C0 Convergence Run	C0 Final ACC (last 5)	C1 Final ACC (last 5)
FreshCo	11	90%	30%
TechParts	11	100%	60%
GreenLeaf	14	100%	70%
QuickShip	9	90%	60%
NordicSteel	21	100%	60%
System (C0)	**14**	**96%**	**56%**

Table 23: Per-client convergence run and final accuracy averaged over the last 5 runs of the 30-run convergence experiment. C0 reaches sustained $\geq$85% accuracy at run 14 across all clients, while C1 plateaus near initial performance.

The convergence trajectory is consistent with the positive feedback dynamics that underpin stigmergic coordination, in which successful assignments amplify the environmental signals that bias future decisions toward the same agent. The Pheromone Engine applies exponential decay (Eq. 1) and reinforcement updates (Eq. 6) to `AffordanceMarker` intensities after each `task-completed` or `task-failed` event. Successful assignments increase marker intensities for the winning agent-task pairing, amplifying that agent's bid score ($B(a_i, \tau)$, Eq. 5) in subsequent rounds.

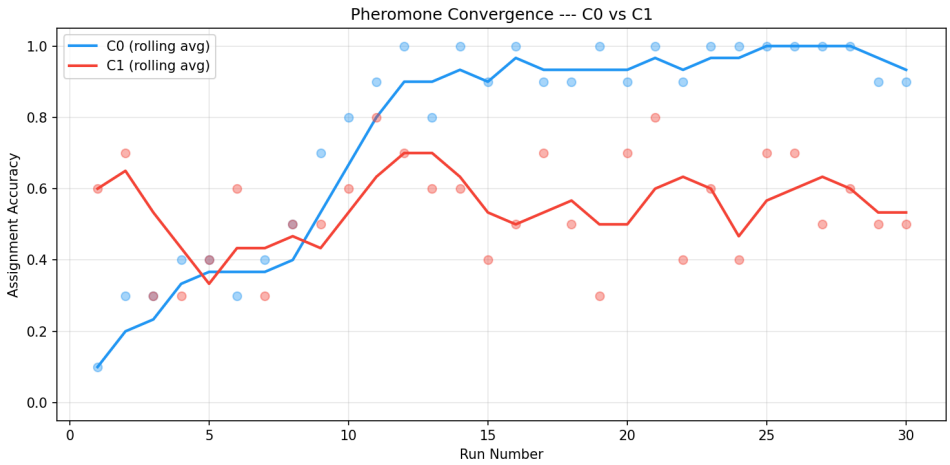


Figure 14: Pheromone convergence over 30 consecutive runs comparing C0 (full framework, markers persist) against C1 (no pheromone, markers cleared). C0 traces the characteristic sigmoid trajectory predicted by stigmergy theory, exceeding 85% at run 14 and saturating near 100%, while C1 remains flat near initial performance.

Over 30 runs, this feedback loop concentrates pheromone mass on the correct agent for each client, producing the characteristic sigmoid trajectory visible in Figure 14. C0 first exceeds 85% accuracy at run 14 and sustains that level, reaching a five-run trailing average of 96% by the end of the experiment ($\Delta = +86$ pp from the 10% observed at run 1), while C1 shows no sustained improvement, finishing at 56% after 30 runs (from 60% at run 1), confirming that without pheromone feedback, performance gains over successive runs are negligible.

Figure 15: `AffordanceMarker` intensity per agent per client over the 30-run convergence experiment under C0. For every client except NordicSteel, the correct agent's intensity separates from competitors by approximately run 10 to 15 and reaches saturation by run 20.

The underlying mechanism is made visible in Figure 15, which plots `AffordanceMarker` intensity per agent per client across all 30 runs. For every client except NordicSteel, the correct agent's intensity separates cleanly from competitors by approximately

run 10 to 15 and reaches saturation by approximately run 20, with incorrect agents decaying toward zero. The NordicSteel panel is a visible exception, as warehouse-north does not begin its sustained rise until around run 10, and its climb to saturation is noticeably shallower and more prolonged than in all other panels. This delayed and gradual separation is consistent with the convergence analysis reported in Table 23 and Figure 14, which confirm that NordicSteel requires 21 runs to reach sustained $\geq$85% accuracy. The initial incorrect assignments generate a competing pheromone signal that the engine must overcome before the correct pattern can dominate.

The progression from near-uniform initialisation to full dominance is further illustrated by the FreshCo snapshots in Figure 16. At run 1, the correct agent's intensity stands at only 0.09. By run 10 it has risen to 0.66, reflecting accumulated reinforcement from the first successful assignments. By run 25 it reaches 1.00, at which point the pheromone landscape is fully consolidated and the correct agent wins every subsequent bid without competition.

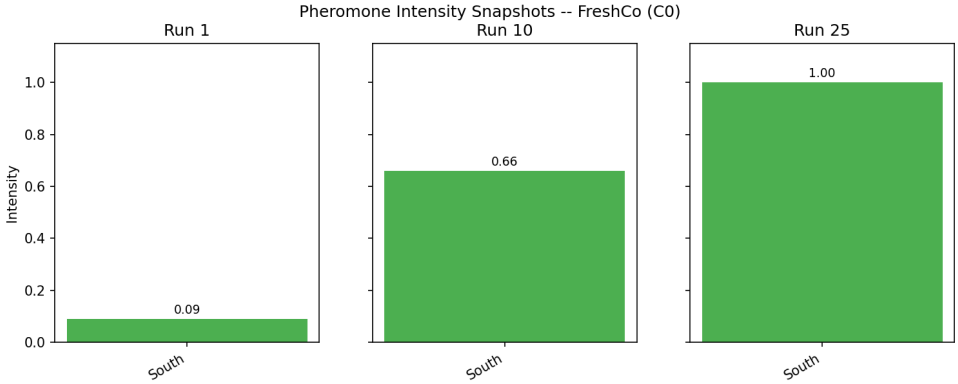


Figure 16: Pheromone intensity snapshots at runs 1, 10, and 25 for the FreshCo client under C0. The progression from near-uniform initialisation (intensity 0.09 at run 1) to full dominance (1.00 at run 25) illustrates the positive feedback dynamic that underpins stigmergic coordination.

The convergence at run 14 is slower than the 5 to 8-iteration convergence reported in robotic swarm systems [80], attributable to two factors. First, stochastic variance in LLM-generated agent outputs produces run-to-run differences in task-completion outcomes for identical agent-task pairs. This noise propagates into pheromone reinforcement (via task-completed/task-failed events) and trust updates, introducing exploration noise that delays exploitation of the correct assignment. Second, the relatively small competing-agent pool (three warehouse agents) limits the reinforcement signal per iteration. Despite this slower convergence, the pheromone mechanism produces a $\Delta = +86$ pp accuracy improvement from initial to converged state in C0, compared to -4 pp in C1 (which drifts from 60% to 56%). The system-level convergence run of 14 reflects the first run at which the rolling average accuracy reaches and sustains $\geq$85% across all clients, rather than the mean of the per-client convergence points, and demonstrates that the pheromone engine transforms noisy per-run decisions into reliable long-term coordination.

The priority-sensitive routing analysis in Subsection 6.2.3 further demonstrates that pheromone-modulated bidding develops context-sensitive behaviour, consistent with stigmergy theory's prediction of globally adaptive behaviour emerging from local reinforcement signals [80, 51].

H1 is therefore supported. The pheromone engine enables emergent learning that transforms initial near-random performance into sustained high accuracy, with system-level convergence at run 14 and a $\Delta = +40$ pp improvement over C1. The observed S-curve and emergent priority-sensitive routing demonstrate that stigmergy can produce

context-adaptive behaviour in LLM-based Multi-Agent Systems, and identify LLM output variance as a parameter that bio-inspired AI coordination theory must explicitly account for, alongside agent pool size and decay rate. The convergence rates reported here are specific to GPT-4o-mini, and a production deployment should calibrate the pheromone parameters (decay constant, reinforcement step) to the variance profile of the selected LLM. The pheromone mechanism's contribution is also invisible in single-run evaluation, as C0 and C1 share identical ACC in the ablation overview (see Subsection 6.2.1), meaning its value can only be observed in a multi-run longitudinal design such as the convergence experiment presented here. Surfacing pheromone dynamics in the system's audit trail, for example by showing how marker intensities evolved toward the selected agent over successive runs, would make this contribution legible to operators without requiring a full longitudinal study.

6.2.3 Priority-Based Agent Selection

Using C0 data alone, this analysis examines how the ICNP's bidding mechanism adapts to task priority metadata, testing the context sensitivity of competitive selection. Table 24 summarises the estimator win ratio by priority level.

Priority	Cost Estimator Wins	Speed Estimator Wins	Total
Critical	0 (0%)	10 (100%)	10
High	22 (73%)	8 (27%)	30
Medium	10 (100%)	0 (0%)	10
Total	32	18	50

Table 24: Estimator selection by client priority level under C0 (N = 50 bidding rounds). Critical-priority tasks route exclusively to the Speed Estimator and medium-priority tasks exclusively to the Cost Estimator, despite no explicit priority rule being encoded in the system.

The results reveal a pronounced routing pattern. At critical priority the Speed Estimator wins every round (100%), while at medium priority the Cost Estimator is selected in every round (100%). High priority produces a majority for the Cost Estimator (73%), with the remaining 27% going to the Speed Estimator, driven by NordicSteel, whose crane installation deadline creates a speed-critical requirement despite the high (rather than critical) priority label. This routing pattern is not encoded anywhere in the system as an explicit rule. It emerges from the interaction between pheromone intensities (which shape both the availability gate and the fit score), trust scores in the bid score B (Eq. 5), and the admission gate S (Eq. 2). The chi-square test of independence confirms the association is statistically significant and large in magnitude ($\chi^2(2) = 24.54$, $p < 10^{-5}$, Cramér's $V = 0.701$), ruling out chance as an explanation for the routing pattern.

The bid score statistics in Table 25 confirm that this routing is the product of genuine competitive discrimination rather than near-random selection. Under C0, the median bid delta between winner and runner-up is 0.22 to 0.23 across both competition points, a non-trivial margin that reflects meaningful score separation. Under C2 and C7, bid deltas are uniformly zero because no bidding occurs, confirming that the margin is a direct product of the ICNP mechanism and not a baseline property of the agents themselves.

The wider bid deltas in C1 relative to C0 carry an additional insight. Without pheromone modulation, all agents start each run with identical baseline marker intensities, producing more polarised bids. Pheromone adaptation in C0 compresses the scoring range as intensities converge toward the correct assignment, explaining the narrower but

Config	WH Bid Delta Mdn (IQR)	EST Bid Delta Mdn (IQR)
C0	0.223 (0.219–0.231)	0.231 (0.230–0.371)
C1	0.263 (0.261–0.275)	0.275 (0.274–0.275)
C2	0.000 (0.000–0.000)	0.000 (0.000–0.000)
C3	0.222 (0.219–0.230)	0.231 (0.230–0.365)
C4	0.221 (0.219–0.230)	0.230 (0.230–0.230)
C3+C4	0.223 (0.218–0.230)	0.230 (0.229–0.231)
C5	0.223 (0.219–0.230)	0.230 (0.230–0.230)
C6	0.223 (0.219–0.230)	0.230 (0.229–0.231)
C7	0.000 (0.000–0.000)	0.000 (0.000–0.000)

Table 25: Bid score delta statistics (winner minus runner-up) by configuration. Non-zero deltas under all ICNP-enabled configurations confirm genuine competitive discrimination, while the zero deltas under C2 and C7 reflect the absence of bidding.

still non-trivial deltas observed. The violin plot in Figure 17 illustrates the warehouse bid score distributions across all three agents under C0. All three agents produce scores in a broadly overlapping range of approximately 0.32 to 0.72, with medians between 0.47 and 0.56. The distributions confirm that bid scores are spread across a meaningful range rather than concentrated at extreme values, consistent with the non-trivial bid deltas reported in Table 25.

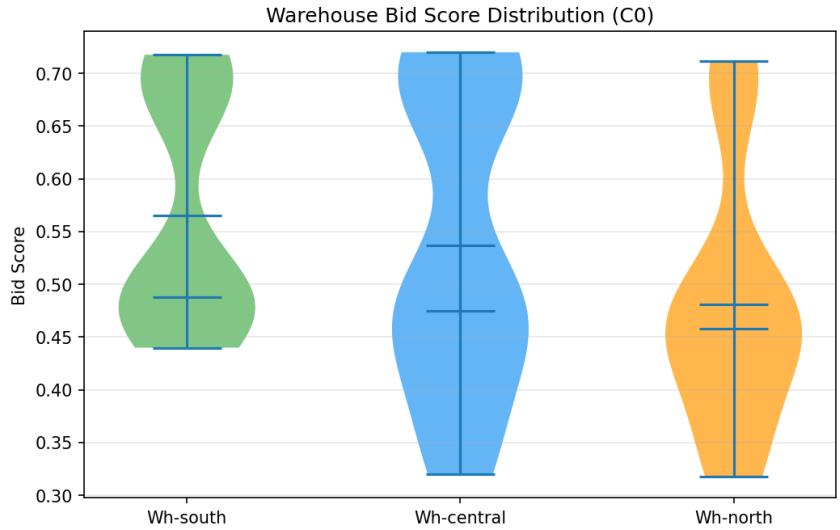


Figure 17: Warehouse bid score distributions per agent under C0. All three agents produce scores in a broadly overlapping range of approximately 0.32 to 0.72, confirming that bid scores are spread across a meaningful range rather than concentrated at extreme values.

H2 is therefore supported. ICNP bidding is the single most critical coordination mechanism ($\Delta = +56$ pp, $\delta = 0.774$, large effect when removed). Under round-robin allocation (C2), warehouse accuracy (34%) and estimator accuracy (50%) match the theoretical random baseline for 3-way and 2-way selection respectively, confirming that the ICNP's score-based selection rather than individual agent LLM quality drives correct assignment. Priority-sensitive routing (Table 24, $\chi^2(2) = 24.54$, $p < 10^{-5}$, Cramér's $V = 0.701$) further confirms that the bidding mechanism adapts to task context as an emergent property of the scoring dynamics rather than as a designed rule. The large effect size for ICNP removal confirms task allocation theory [1, 65], in that market-based mechanisms outperform static policies whenever agents have non-uniform capabilities. The result holds here

even with a small agent pool ($N = 3$ warehouse agents, $N = 2$ estimators) and a single-domain evaluation, suggesting the effect is robust rather than contingent on scale. More broadly, the finding implies that any Multi-Agent orchestration system in which multiple agents share a capability but differ in quality must use a competitive selection mechanism or accept near-random allocation as the alternative. Individual agent quality is necessary but not sufficient. The coordination mechanism that adjudicates between agents is what determines system-level performance. The ICNP also carries an inherent transparency property worth preserving in future framework variants, as every allocation decision is backed by a scored bid and the system logs the winning score, runner-up score, and each agent's capability alignment rationale as a natural by-product of coordination, producing auditable decisions not as a separate explainability layer but as a direct consequence of how the selection mechanism works.

6.2.4 Validation Layer Interaction

To test **H3** (super-additive defence-in-depth), this analysis examines whether the SHACL and compliance validation layers jointly reduce downstream error propagation more than either layer alone. The evidence draws on three complementary sources. The first is the ablation pipeline runs, specifically the 300 runs across the six configurations where the Compliance Agent is enabled (C0, C1, C2, C3, C5, C6) out of the 450 total ablation runs. Table 10 defines the ground-truth violation status per client, yielding a ground-truth violation rate of $2/5 = 40\,\%$. The second is a targeted validation probe consisting of ten synthetic test cases run outside the main pipeline with no LLM calls, in which six structural defects, three domain defects, and one negative control were injected into plan drafts to verify that each validation layer detects the defect classes it is designed for. The third is a 2×2 factorial analysis across the four validation configurations (C0, C3, C4, C3+C4), computing an interaction effect on assignment accuracy to test for super-additivity. Table 26 reports the compliance check outcomes per client across configurations where the Compliance Agent is enabled.

Client	Runs	Passed	Avg Blocks	Avg Warnings
FreshCo	60	0/60	2.0	4.0
TechParts	60	58/60	0.2	3.3
GreenLeaf	60	0/60	5.5	4.6
QuickShip	60	60/60	0.0	0.8
NordicSteel	60	60/60	0.0	2.6

Table 26: Compliance check results per client across the six configurations where the Compliance Agent is active (60 runs per client). FreshCo and GreenLeaf are seeded to fail by design, while the two TechParts blocks are isolated to runs under C2 and analysed in the surrounding text.

FreshCo and GreenLeaf fail compliance in all 60 runs by design. Each is paired with one or more warehouses seeded to violate a deterministic rule, as enumerated in Table 10. These detections are engineered ground-truth positives used to measure the agent's recall rather than stochastic compliance failures. The per-run block counts, averaging 2.0 for FreshCo and 5.5 for GreenLeaf, reflect the number of injected violating pairs in scope on each run.

TechParts passes 58 of 60 runs. The two BLOCK-emitting runs are rounds 4 and 10 of C2, where round-robin allocation bypasses ICNP and routes the client through the speed-estimator instead of the cost-estimator. The resulting plans trigger the deterministic compliance rules even though the warehouse assignment itself remains correct, with

warehouse-central selected in both runs, isolating the false-positive source as degraded estimator quality propagating into the validation layer rather than warehouse misassignment. The speed-estimator is substituted in all 10 C2 runs yet only 2 emit blocks, meaning the substitution is necessary but not sufficient, and the 2 of 10 incidence reflects residual LLM stochasticity in plan content even at temperature 0. NordicSteel and QuickShip pass all 60 runs, confirming that the Compliance Agent produces no spurious blocks for clients whose requirements are satisfiable under coordinated assignment.

The full classification performance of the Compliance Agent across all 300 client-warehouse pairings is reported in Table 27. Perfect recall (TPR = 1.000) follows directly from deterministic enumeration over the rule set on each scoped pair rather than probabilistic inference, providing the formal guarantees that regulated domains require.

	Predicted Violation	Predicted Clean	
Actual Violation	TP = 120	FN = 0	
Actual Clean	FP = 2	TN = 178	
TPR (Recall) = 1.000	FPR = 0.011	Precision = 0.984	F1 = 0.992

Table 27: Compliance validation confusion matrix across all 300 client and warehouse pairings. Perfect recall (TPR = 1.000) follows from deterministic enumeration of the rule set rather than probabilistic inference.

To verify that SHACL and the Compliance Agent cover non-overlapping defect classes, the targeted probe injected six structural defects, three domain defects, and one negative control into plan drafts. Table 28 records the result. SHACL detects all six structural defects and none of the domain defects. The Compliance Agent detects all three domain defects and none of the structural defects. D4, a QuickShip-to-Lyon pairing, is a valid assignment by domain rules and correctly produces no alert from either layer, confirming that the Compliance Agent does not over-fire on clean inputs.

Defect	Class	SHACL	Compliance
S1: Missing `assigned_agent`	Structural	✓	—
S2: Empty capabilities list	Structural	✓	—
S3: Missing task description	Structural	✓	—
S4: Hazmat task without human approval	Structural	✓	—
S5: Cold-storage task without temp range	Structural	✓	—
S6: Critical deal without high-risk flag	Structural	✓	—
D1: FreshCo → Bavaria Hub (no cold storage)	Domain	—	✓
D2: GreenLeaf → Bologna (no hazmat, std. security)	Domain	—	✓
D3: GreenLeaf → LogisPark Milano (no hazmat)	Domain	—	✓
D4: QuickShip → Lyon (valid pairing)	Control	—	—
Detection summary		6/6 structural	3/3 domain

Table 28: Validation probe detection matrix. SHACL detects all 6 of 6 structural defects and the Compliance Agent detects all 3 of 3 domain defects, with zero overlap between the two layers, confirming complementary rather than redundant coverage.

Figure 18 shows how detected violation rate varies across the four ablation configurations. The drop to 0 % in C4 and C3+C4 is not a reduction in violations but an absence

of the detection mechanism, as the ground-truth violation rate remains 40 % regardless of which configuration is active. Figure 19 visualises the zero-overlap result from the probe, confirming that SHACL and the Compliance Agent occupy entirely separate regions of the defect space.

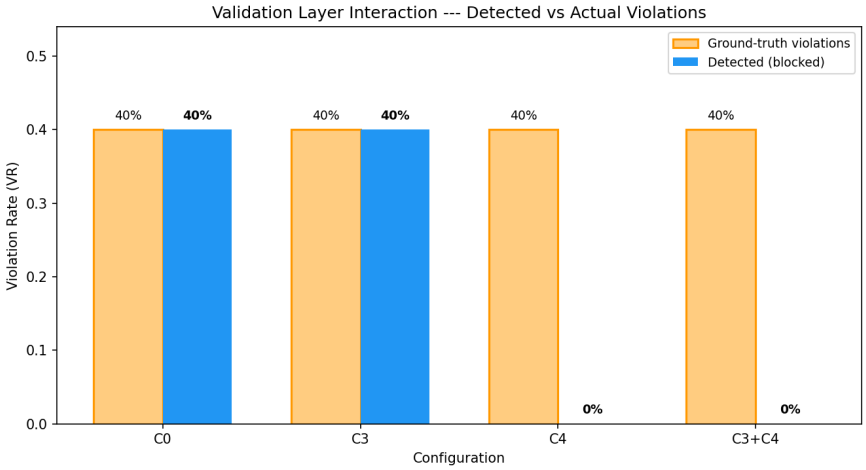


Figure 18: Detected violation rate against the 40% ground-truth floor by configuration. The drop to 0% in C4 and C3+C4 reflects the absence of the detection mechanism rather than a reduction in actual violations.

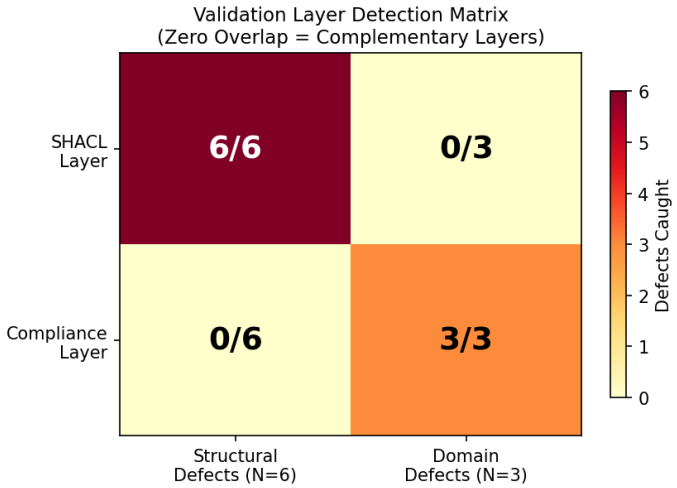


Figure 19: Validation layer detection heatmap. SHACL covers all 6 of 6 structural defects and the Compliance Agent covers all 3 of 3 domain defects, with zero overlap between layers, confirming complementary rather than redundant coverage.

To test whether the two layers interact super-additively, the four validation configurations are arranged as a 2×2 factorial design (Table 29), with SHACL (on/off) and Compliance (on/off) as the two factors and assignment accuracy as the response variable. ON denotes the component is active in the pipeline DAG and OFF denotes it is removed entirely, consistent with the ablation definitions in Table 17. The interaction effect is computed on accuracy rather than on the detected violation rate, because violation rate drops to 0 % when the compliance detector is absent, a tautological artefact that cannot support interaction analysis.

The interaction effect on accuracy is computed as follows.

$$\text{IE} = \text{ACC(C0)} - \text{ACC(C3)} - \text{ACC(C4)} + \text{ACC(C3+C4)} = 98 - 85 - 90 + 98 = +21\,\text{pp}$$

	Compliance ON	Compliance OFF
SHACL ON	C0 = 0.98	C4 = 0.90
SHACL OFF	C3 = 0.85	C3+C4 = 0.98

Table 29: 2×2 factorial design for validation layer interaction analysis, with SHACL and Compliance as factors and assignment accuracy as the response variable. The diagonal cells (C0 and C3+C4) sit above the off-diagonal cells (C3 and C4), the signature of a super-additive interaction.

Each layer alone degrades accuracy relative to the no-validation baseline (C3+C4 = 0.98). Removing compliance only (C4) costs −8 pp and removing SHACL only (C3) costs −13 pp, yielding a predicted additive penalty of −21 pp. The observed penalty when both layers are present (C0) is 0 pp, producing a super-additive interaction of +21 pp. Figure 20 visualises this crossing interaction pattern. The non-parallel lines confirm that the joint effect exceeds the sum of the marginal effects, with the observed C0 accuracy of 0.98 exceeding the additive prediction of 0.77 by 21 percentage points.

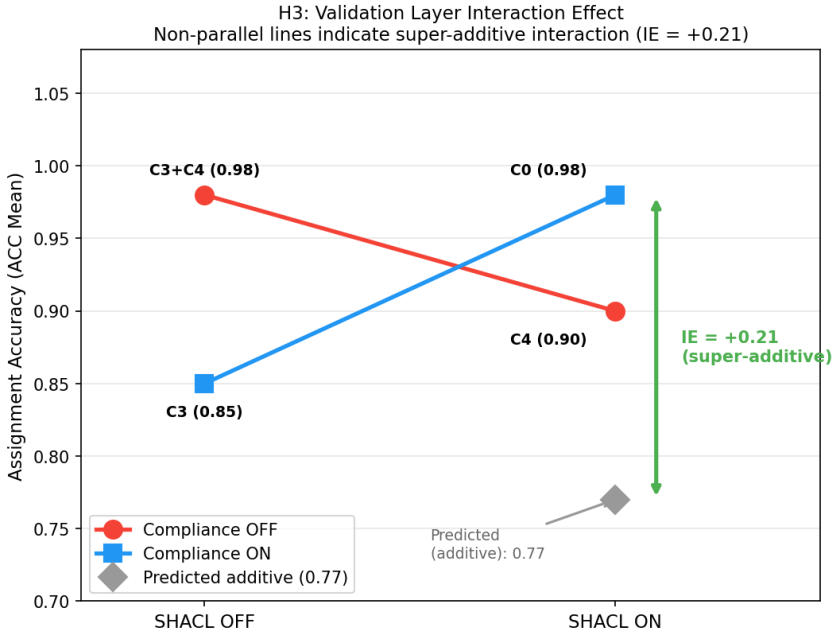


Figure 20: Validation layer interaction plot from the 2×2 factorial design. The crossing, non-parallel lines indicate a super-additive interaction (IE = +21 pp), with observed C0 accuracy of 0.98 exceeding the additive prediction of 0.77.

The Friedman test across the four configurations yields $\chi^2 = 7.36$, $p = 0.0612$, which does not reach significance at $\alpha = 0.05$, reflecting low statistical power with only four conditions and five paired observations rather than an absence of a meaningful effect. The interaction effect size (IE = +21 pp) is therefore the more informative quantity.

The mechanistic explanation for the super-additive effect lies in the zero-overlap detection pattern documented in Table 28. SHACL enforces the structural contract of the plan representation, verifying that required fields exist, types are correct, and structural dependencies are satisfied, independently of what the plan says. The Compliance Agent enforces the domain contract, evaluating whether the plan's semantic content satisfies business rules, independently of whether the plan is well-formed. When only one layer operates, the uncovered defect class propagates downstream and interferes with the pipeline's assignment logic, degrading accuracy. SHACL alone catches structural violations and forces plan revision, but the revised plan may still carry domain violations

that accumulate silently, producing assignment instability. The Compliance Agent alone catches domain violations and blocks certain pairings, but structurally malformed plans that pass compliance scoping checks proceed uncorrected into downstream stages. Only when both layers operate simultaneously is each defect class resolved at its appropriate abstraction level, eliminating cross-class error leakage and restoring accuracy to the coordination baseline. A plan can be structurally valid, with SHACL confirming conformance in 100 % of runs where it is enabled, while simultaneously violating domain rules in 40 % of those runs. These two contracts are logically disjoint and satisfying one neither implies nor precludes satisfying the other. The zero-overlap detection pattern is therefore not a logistics artefact but a structural consequence of layered abstraction that generalises to any domain instantiation of the framework, including medical treatment plans, financial transactions, legal contracts, and supply chain orders.

H3 is therefore supported. The 2×2 factorial analysis reveals a super-additive interaction effect of +21 pp on assignment accuracy. Each validation layer alone degrades accuracy from the no-validation baseline, but the combination restores it fully, producing a safety improvement greater than the sum of its parts. The probe detection matrix confirms the mechanism, with SHACL catching all 6/6 structural defects and the Compliance Agent catching all 3/3 domain defects with zero detection overlap, and the Compliance Agent achieving TPR = 1.000 and F1 = 0.992 across 300 evaluated pairings. The practical implication for AI system design is that safety mechanisms should be stratified by abstraction level rather than duplicated at a single level. A system with two structural validators gains nothing over one, whereas a system with one structural and one domain validator covers qualitatively different failure classes and provides error-propagation resilience that neither layer achieves in isolation. This also closes the loop on the C3+C4 result flagged earlier. The ICNP still selects the correct agent without either validation layer, but the pipeline loses all visibility into constraint violations. A system that selects the correct warehouse while failing to detect a cold-storage violation is accurate but unsafe, which is why any evaluation reporting only assignment accuracy will miss safety-layer failures.

6.2.5 Failure Recovery

To evaluate the multi-tier recovery mechanism (Section 4.9), agent failures are injected after task assignment but before execution completion across C0, C1, C2, and C7 ($N = 10$ per configuration). RSR denotes Recovery Success Rate and RT denotes Recovery Time in seconds. Table 30 reports the results.

Config	Runs with Failure	Recovered	RSR	RT Mdn (s)	RT IQR (s)
C0 (full)	10	10	100%	55	42–85
C1 (−pheromone)	10	10	100%	48	38–80
C2 (−ICNP)	10	3	30%	153	88–176
C7 (minimal)	10	0	0%	—	—

Table 30: Failure recovery results across four configurations ($N = 10$ injections per configuration). Recovery is reliable when both the bounded retry layer and the LLM Strategist's adaptive replanning are present (C0, C1), degrades sharply once ICNP is removed (C2), and disappears entirely under the minimal baseline (C7).

The results separate into two qualitatively distinct recovery regimes. Both C0 and C1 achieve a 100% Recovery Success Rate, with median recovery times of 55 s and 48 s respectively and overlapping interquartile ranges. The two configurations share the full Tier 1 and Tier 2 machinery defined in Algorithm 6, namely bounded local retries followed

by adaptive replanning through the LLM Strategist (Section 4.8), which proposes graph deltas that re-route failed tasks to alternative agents. The fact that recovery succeeds in every injected run for both configurations indicates that the combination of bounded retry and validated graph-delta replanning is sufficient to absorb the injected failures without escalation to Tier 3.

The 7 s gap between C1 and C0 in median recovery time reflects the absence of the pheromone penalty step in C1. In C0, every failure event triggers a penalty update to the failed agent's marker intensity (Eq. 6), executed concurrently with the Tier 2 replanning action. C1 omits this step entirely because the pheromone engine is disabled. The cost of the penalty write therefore appears as a small additive overhead in C0's recovery latency. Crucially, this overhead also represents the only mechanism in the framework that durably steers future assignments away from an unreliable agent. C1 recovers each failure as it arrives but retains no learned signal, so subsequent runs remain equally likely to route to the previously failed agent. C0 trades a few seconds of per-event latency for a self-correcting behaviour in which each failure progressively biases the bidding process away from the responsible agent.

C2 recovers only three of ten injected failures, with a median recovery time of 153 s and an interquartile range that extends close to the 300 s timeout. The collapse in RSR from 100% to 30% follows from the removal of the ICNP, which the framework relies on to produce the candidate set used by the Tier 2 replanner. Without ICNP-driven selection, re-routing cannot draw on a ranked alternative for the failed agent, and recovery succeeds only when the fallback policy happens to reach an eligible agent within the timeout window. The widened latency distribution is consistent with this account, since each successful recovery in C2 requires re-issuing the task through a less directed selection process rather than promoting a pre-validated alternative.

C7, the minimal configuration with all coordination components removed, recovers none of the ten injected failures. With no retry middleware, no Strategist, and no escalation pipeline, every injection propagates unhandled and terminates the run. This establishes the lower bound for recovery in the absence of any framework mechanism and confirms that automatic failure recovery is a property of the coordination layer introduced in this work rather than of the underlying agent runtime.

Appendix B traces one C0 injection step-by-step, showing the three-tier mechanism activate from `task-failed` detection through Tier 2 adaptive replanning, with the pheromone penalty applied as a concurrent Tier 2 action and Tier 3 escalation not reached. The total recovery time of 62 s is consistent with the C0 distribution reported in Table 30 and the 100% RSR observed across all C0 injections.

Taken together, these results indicate that the three-tier recovery mechanism delivers reliable failure resolution when both the bounded retry layer and the LLM Strategist's validated replanning are present, and that recovery degrades sharply once the ICNP is removed and disappears entirely once all coordination is stripped out. The pheromone penalty does not change recovery outcomes within a single run but accounts for a small per-event latency cost in exchange for a durable learning signal that biases subsequent assignments away from agents implicated in past failures.

6.3 Qualitative Analysis

This section presents the results of a user perception survey conducted to evaluate the proposed Multi-Agent framework from the perspective of prospective end users. The evaluation follows a scenario-based design in which participants reviewed pre-recorded

outputs from the system across three warehouse brokerage scenarios of varying complexity. The analysis is structured around four dimensions. These are output quality, decision transparency, framework value, and comparative judgment. Free-text responses are analysed through thematic coding to surface recurring concerns, expectations, and points of consensus across participant groups.

6.3.1 Survey Instrument

The evaluation instrument, reproduced in Appendix E, was deployed as an online survey following a within-subjects design. All participants reviewed the same set of three scenario packets produced by the proposed framework.

- **Scenario 1 (FreshCo)** - represents a straightforward cold-storage warehouse request in Milan, reflecting a routine brokerage task with clear requirements and a well-matched warehouse in the system's inventory.
- **Scenario 2 (GreenLeaf)** - represents a hazardous-materials pharmaceutical warehousing request with stricter regulatory constraints, reflecting a compliance-sensitive case.
- **Scenario 3 (Failure Recovery)** - represents a request deliberately configured to trigger the system's failure-handling pipeline, where no fully eligible warehouse exists in the target region.

Each scenario packet included the client request, the supervisor agent's task decomposition, the ICNP bidding summary, the warehouse recommendation, compliance check results, the generated client proposal, and the decision provenance trace. Section 4 of the survey additionally presented three side-by-side outputs for comparative judgment, namely Output A from a baseline LLM, Output B from a system failure case, and Output C from the full framework. The full text of the three outputs, together with the shared client request, is reproduced in Appendix E.

The survey comprised 31 questions across four thematic sections, preceded by four GDPR consent checks. The sections covered output quality through 10 Likert items, decision transparency through 4 Likert items, framework value through 4 Likert items administered to technical evaluators only, and comparative judgment through 2 forced-choice items and 1 ranking item, supplemented by six open-ended questions and a three-item participant profile. All Likert items used a 5-point scale ranging from 1 for Strongly Disagree to 5 for Strongly Agree.

6.3.2 Participant Sample

In total, 19 participants completed the survey. Participants were recruited through purposive sampling and assigned to one of two evaluator groups based on their professional background:

- **Domain experts** ($n = 9$): Professionals in logistics, warehouse brokerage, or supply chain management.
- **Technical evaluators** ($n = 10$): Software engineers, AI/ML practitioners, or researchers with experience in Multi-Agent or LLM-based systems.

Table 31 summarises the experience distribution across the sample. AI familiarity varied considerably, with 7 participants (36.8%) identifying as "very familiar" in the sense of building or researching AI systems, 5 (26.3%) as "familiar" through regular professional use, 5 (26.3%) as "somewhat familiar" through casual use, and 2 (10.5%) as "not familiar at all." This distribution ensures that the evaluation captures perspectives from both participants with strong AI familiarity and those with limited or no prior exposure, reflecting the mixed-expertise teams common in logistics organisations.

Years of Experience	**Count**	**Percentage**
Less than 2 years	3	15.8%
2–5 years	4	21.1%
5–10 years	7	36.8%
More than 10 years	5	26.3%

Table 31: Participant experience distribution ($N = 19$).

6.3.3 Output Quality

This subsection reports participant ratings for the system's output quality across the three evaluation scenarios. Table 32 summarises the descriptive statistics for all output quality items, disaggregated by evaluator group, and Figure 21 presents the mean ratings visually.

Item	Mdn	M	SD	IQR	M_D	M_T
Scenario 1: FreshCo						
Warehouse match	5	4.47	0.61	4–5	4.33	4.60
Compliance check	4	4.32	0.48	4–5	4.00	4.60
Proposal quality	4	3.68	0.58	3–4	3.22	4.10
Novel information	4	3.63	0.60	3–4	3.44	3.80
Scenario 2: GreenLeaf						
Warehouse match	4	3.68	0.58	3–4	3.44	3.90
Compliance check	4	4.11	0.57	4–4	4.00	4.20
Proposal quality	3	3.37	0.50	3–4	3.11	3.60
Novel information	4	3.58	0.61	3–4	3.33	3.80
Scenario 3: Failure Recovery						
Failure handled	4	3.58	0.77	3–4	3.22	3.90
Fallback recommendation	3	2.84	0.76	2–3	2.44	3.20

Table 32: Output quality ratings across the three evaluation scenarios ($N = 19$). M_D is the domain expert mean and M_T the technical evaluator mean.

The FreshCo scenario received the highest ratings overall (Table 32). Warehouse suggestion accuracy and compliance checking both scored well above the neutral midpoint, with 18 of 19 participants rating warehouse match at 4 or above and all 19 scoring compliance at 4 or 5. These results suggest that the Multi-Agent pipeline produces reliable core outputs for well-defined, routine tasks. Proposal quality and novelty of information received moderately positive ratings but revealed the first notable group divergence. Domain experts rated proposal quality significantly lower than technical evaluators (large effect, see Table 36), indicating that logistics professionals hold higher expectations for client-facing documents than the engineers testing the system.

The GreenLeaf scenario, involving hazardous materials and pharmaceutical GDP requirements, produced lower ratings for warehouse matching accuracy compared to

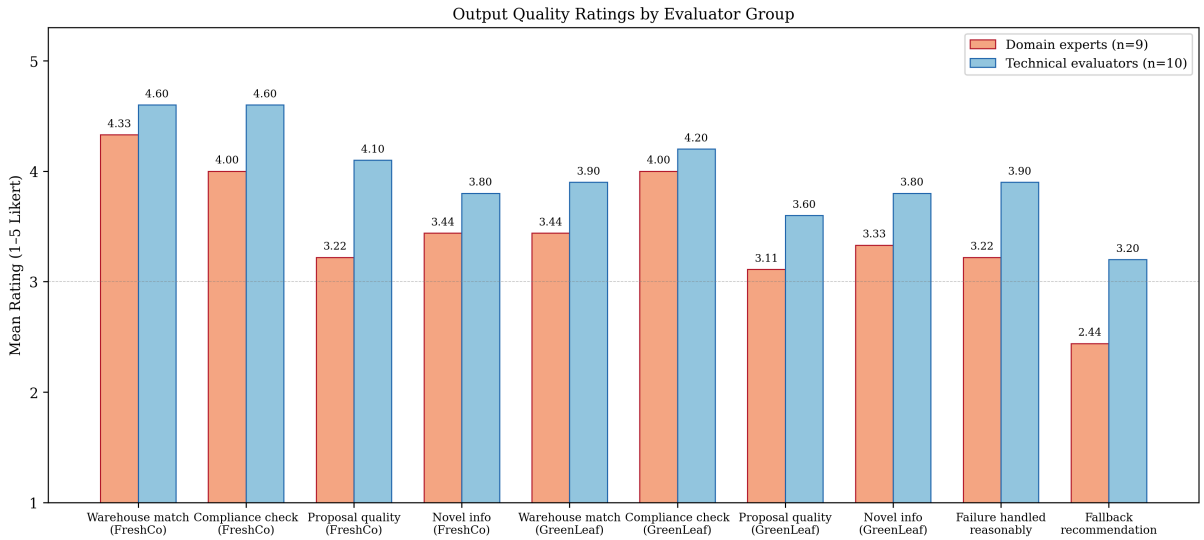


Figure 21: Output quality ratings by evaluator group across all scenarios ($N = 19$). The dashed line indicates the neutral midpoint.

FreshCo. This reduction is expected given the tighter regulatory constraints and fewer eligible warehouses. Compliance checking remained robust, confirming that the validation layer performs well even under stricter conditions. Proposal quality dropped further, with 12 of 19 participants rating it neutral and the domain–technical gap again statistically significant (medium effect, Table 36). This confirms that for high-stakes, compliance-sensitive cases, the system's generated proposals require substantial human refinement before client delivery.

The failure recovery scenario assessed participants' perception of the system's behaviour when no fully eligible warehouse exists. Failure handling was rated as reasonable but not strong, while the fallback recommendation received the lowest rating in the entire survey, with 7 of 19 participants scoring it at 2 (Disagree). The group gap was again significant (medium effect; Table 36). This highlights a clear improvement area: while the framework's multi-tier failure handling mechanism (Section 4.9) successfully surfaces the failure and escalates it, the fallback recommendations themselves lack the actionable specificity that domain experts expect. As participant P13[D] noted, *"the failure recovery output was not very actionable."* The system exposes the problem transparently, which several participants valued (P15[T]: *"transparent failure handling is a strength"*), but the recovery content requires enrichment. This gap is largely a function of the underlying language model and prompt scaffolding used to generate recovery suggestions. A stronger or domain-adapted LLM, combined with prompts that explicitly request concrete next steps and alternative options, would substantially improve the actionability of fallback recommendations without requiring changes to the failure-handling architecture itself.

6.3.4 Decision Transparency

Decision transparency was assessed through four items measuring participants' ability to understand, trace, and audit the system's decisions. Figure 22 presents the results.

All four transparency items received positive overall ratings (Table 33). Understanding why a warehouse was selected and understanding the bidding scores were rated highest, indicating that the ICNP bidding mechanism and its structured audit trail successfully communicate the system's reasoning.

The group divergence is most uniform in this subsection, with all four items crossing

the large-effect threshold. Technical evaluators consistently rated transparency items near the ceiling, while domain experts scored them notably lower. Mann–Whitney U tests confirmed that all four items differed significantly between groups, all with large effect sizes (Table 36). This pattern suggests that while the transparency mechanisms are structurally sound, their presentation may be overly technical for non-engineering audiences. As P16[D] observed, *"I can tell the system is trying to be transparent, but it still feels hard to read."* Several domain experts (P2[D], P9[D], P12[D]) echoed this concern, requesting simpler explanations and clearer business summaries.

Error localisation received the lowest transparency score and showed the largest group gap, indicating that identifying where an error occurred in the pipeline is intuitive for those familiar with Multi-Agent architectures but challenging for domain practitioners.

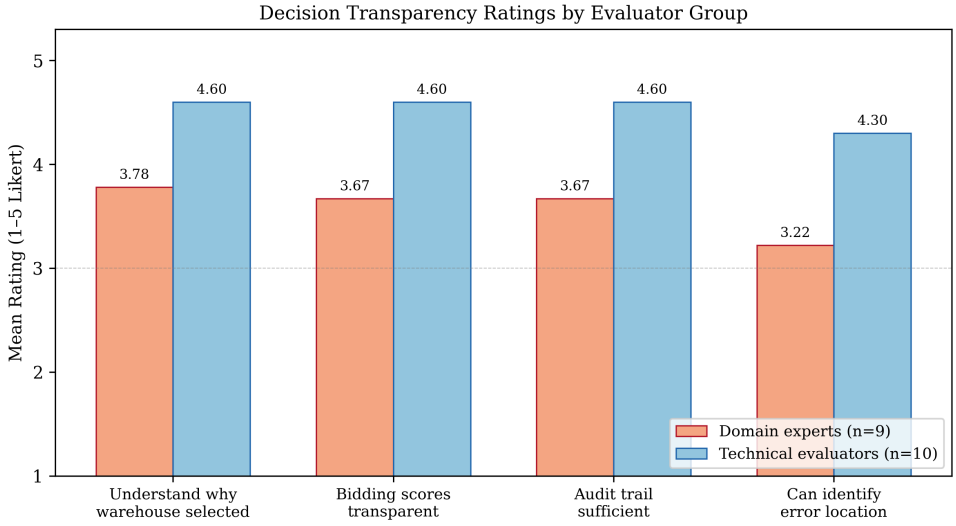


Figure 22: Decision transparency ratings by evaluator group ($N = 19$). Technical evaluators consistently rate transparency items near the ceiling, while domain experts score them notably lower across all four items.

Item	Mdn	M	SD	IQR	M_D	M_T
Understand why selected	4	4.21	0.63	4–5	3.78	4.60
Bidding scores transparent	4	4.16	0.69	4–5	3.67	4.60
Audit trail sufficient	4	4.16	0.76	4–5	3.67	4.60
Can identify error location	4	3.79	0.79	3–4	3.22	4.30

Table 33: Decision transparency ratings ($N = 19$). M_D is the domain expert mean and M_T the technical evaluator mean. All four items differ significantly between groups with large effect sizes (see Table 36).

6.3.5 Framework Value Assessment

Four items assessed technical evaluators' ($n = 10$) perceptions of the framework's architectural contributions. These items were restricted to the technical group, as they require familiarity with alternative system designs. Figure 23 presents the results.

Table 34 summarises the framework value ratings. The Multi-Agent approach received strong endorsement over a monolithic LLM design, with all 10 technical evaluators rating it 4 or 5. Failure recovery mechanisms were similarly valued. As P7[T] noted, *"this is exactly where Multi-Agent design beats a monolithic call. Specialisation, checks, recovery, and traceability."*

SHACL-based validation received moderately positive ratings, with 8 of 10 participants rating it 4 or above. The pheromone-based adaptation mechanism was the most contested component, with ratings distributed across the full range (2–4). Several participants expressed difficulty justifying the mechanism operationally. P17[T] stated that *"pheromone scoring is the least convincing part,"* while P6[T] noted that *"the pheromone part may be hard to defend to non-technical teams."* P3[T] added that *"pheromone logic feels harder to justify operationally."* This consistent scepticism suggests that while the bio-inspired coordination mechanism is theoretically grounded (Section 2.4), its practical value proposition requires stronger empirical demonstration than the current prototype provides.

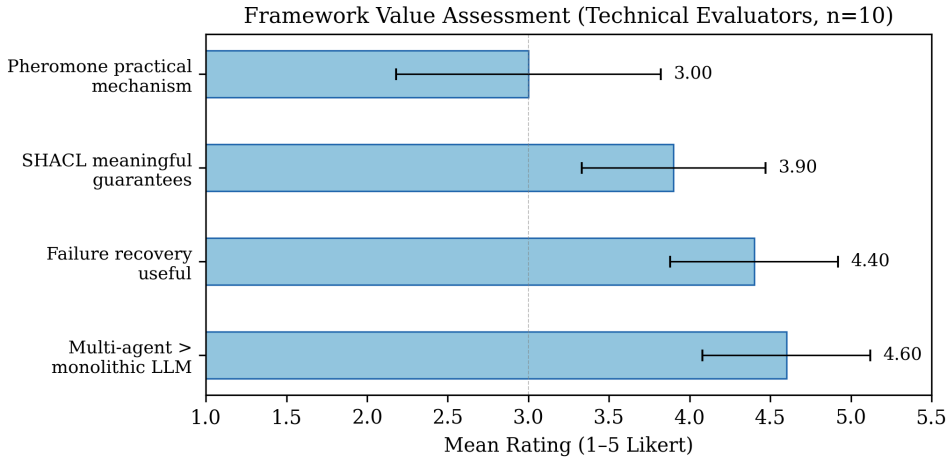


Figure 23: Framework value ratings from the technical evaluator subgroup ($n = 10$), with error bars indicating one standard deviation. The Multi-Agent and failure recovery items attract narrow consensus near the ceiling, while the pheromone mechanism shows the widest spread.

Item	Mdn	M	SD	Distribution
Multi-Agent > monolithic LLM	5	4.60	0.52	4: 4, 5: 6
Failure recovery useful	4	4.40	0.52	4: 6, 5: 4
SHACL meaningful guarantees	4	3.90	0.57	3: 2, 4: 7, 5: 1
Pheromone practical mechanism	3	3.00	0.82	2: 3, 3: 4, 4: 3

Table 34: Framework value ratings from the technical evaluator subgroup ($n = 10$). The Multi-Agent design and failure recovery mechanisms attract strong consensus, SHACL validation moderate support, and the pheromone mechanism the most divided ratings.

6.3.6 Comparative Judgment

Participants also compared three side-by-side outputs representing different system configurations: Output A (a baseline LLM response without the framework), Output B (the failure case with visible error handling), and Output C (the full framework output with structured reasoning, bidding traces, and compliance checks). Figure 24 presents the forced-choice results for trust and audit trail quality, and Figure 25 presents the full ranking distribution.

Output C (full framework) was selected as the most trusted by 84% of participants (16/19) and as the best for audit purposes by 95% (18/19). No participant selected Output B for either question. Three participants preferred Output A for trust, all from the domain expert group, suggesting that some practitioners prioritise simplicity and

readability over structured traceability. In the full ranking, Output C was ranked first by 16 of 19 participants (84%), Output B was ranked last by 15 of 19 (79%), and Output A occupied a consistent middle position, receiving Rank 2 from 12 participants (63%). These results validate the framework's value proposition, as structured Multi-Agent outputs with visible reasoning are consistently preferred over both unstructured baselines and failure cases.

Participants' explanations for their preferences revealed two consistent patterns. First, *explainability drives preference*: P1[T] stated *"I preferred C because it explains the reasoning instead of just giving an answer,"* and P10[T] noted that C *"shows the reasoning instead of just asserting it."* Second, *defensibility matters for domain experts*: P11[D] preferred C because *"I could actually defend that recommendation internally,"* and P12[D] chose it because *"it feels closest to how a serious brokerage team would justify a recommendation."* Two domain experts (P2[D], P16[D]) preferred Output A, citing simplicity and readability, a minority view that nonetheless highlights the need for configurable output verbosity.

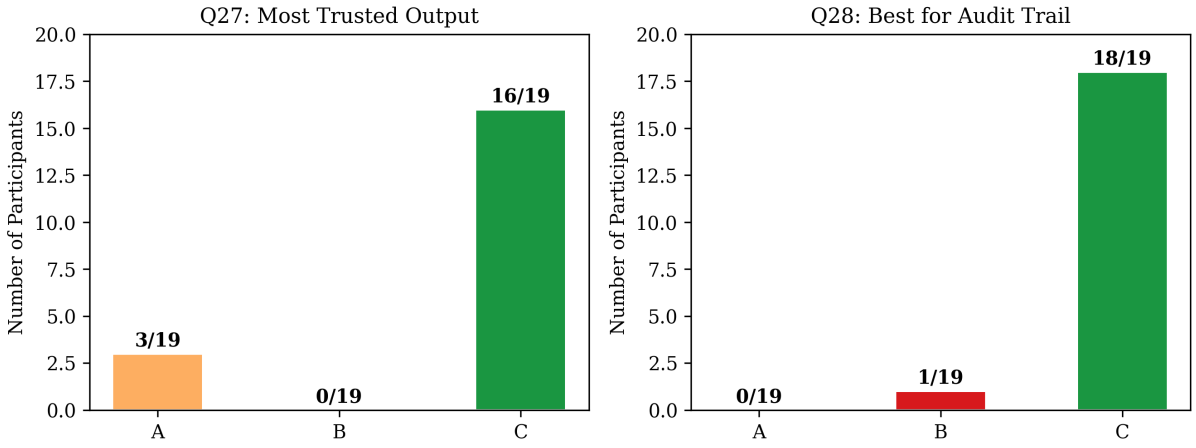


Figure 24: Forced-choice preferences for trust (Q27) and audit trail quality (Q28), $N = 19$. Output C (full framework) was selected as the most trusted by 84% of participants and as the best for audit by 95%.

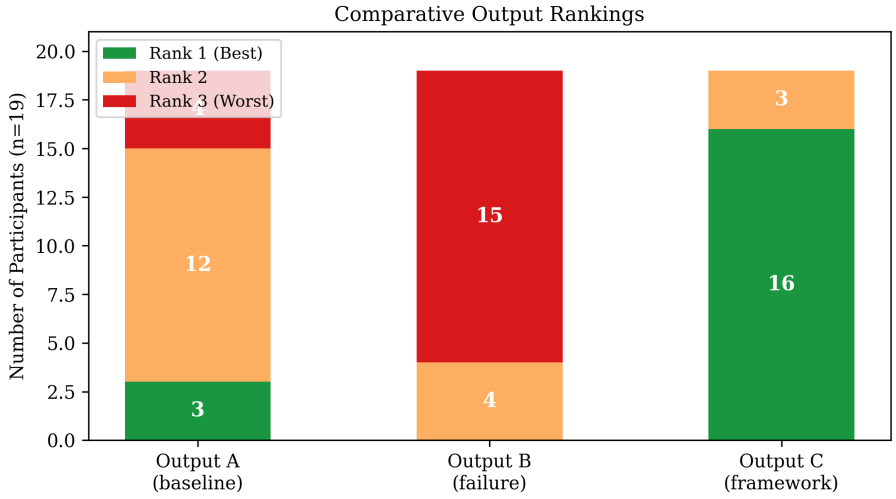


Figure 25: Comparative output rankings ($N = 19$). Each participant ranked all three outputs from best (Rank 1) to worst (Rank 3).

Overall system usefulness was rated positively, with a statistically significant difference

between groups at a large effect size as reported in Table 36, where technical evaluators rated usefulness notably higher than domain experts. Figure 26 illustrates this comparison.

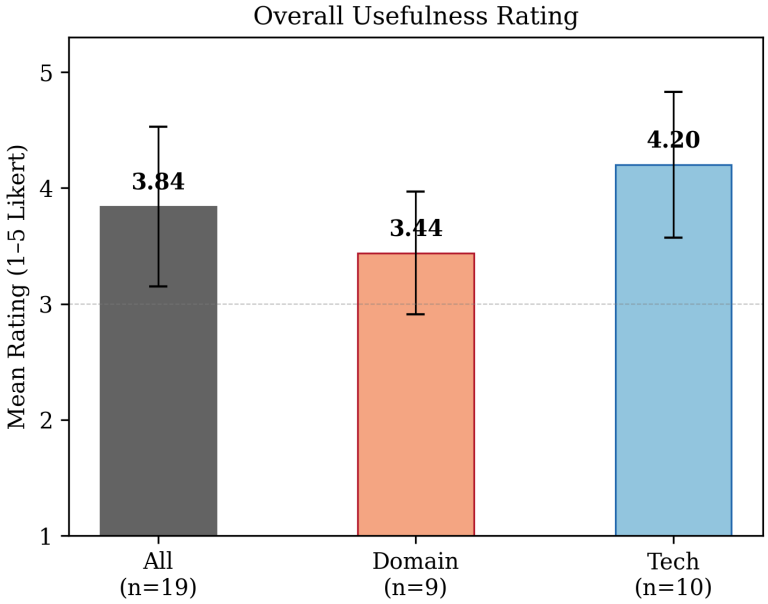


Figure 26: Overall usefulness ratings by evaluator group ($N = 19$). Technical evaluators rated the system notably higher than domain experts.

6.3.7 Thematic Analysis of Open-Ended Responses

Six open-ended questions elicited free-text responses, five answered by all 19 participants and one restricted to the technical group. Responses were analysed using inductive thematic coding, where each response was segmented into discrete observations, grouped by semantic similarity, and consolidated into themes. A theme was retained if it appeared in at least three independent responses. Six themes emerged, summarised in Table 35.

Theme	Freq.	Group(s)	Design Implication
Evidence provenance	8/19	Tech (1 D)	Surface provenance links in user-facing outputs
Conditional trust	14/19	Both	System correctly positioned as decision support, not replacement
Accessibility gap	6/19	Domain	Add configurable output verbosity and executive summaries
Market intelligence	6/19	Domain	Integrate external market data via additional agents
Pheromone scepticism	5/10	Tech	Strengthen empirical validation or simplify mechanism
Failure transparency	4/19	Tech	Validate as design differentiator and extend to non-technical outputs

Table 35: Summary of the six themes that emerged from the inductive thematic coding of open-ended responses. Themes were retained when they appeared in at least three independent responses.

The most frequently cited gap was the absence of verifiable evidence links in the system's outputs. Eight of 19 participants (42%) explicitly requested source links, evi-

dence URLs, or direct references to certification databases, across both evaluator groups. Technical evaluators emphasised machine-readable provenance: P3[T] noted *"missing evidence links, especially for compliance and certification assumptions,"* while P7[T] requested *"clickable links from each claim back to source data."* Domain experts focused on practical implications: P8[D] stated that *"for GreenLeaf I would want actual proof of GDP status, not just a note to verify it."* This finding directly relates to the framework's provenance modelling described in Section 4.10, where the system records PROV-inspired `prov:wasGeneratedBy` relations internally but does not surface them in user-facing outputs. Three further themes emerged in Q15 specifically (Figure 27), all concentrated among technical evaluators, namely explicit confidence or uncertainty estimates reported by 5 of 19 participants, sensitivity-to-assumption analysis reported by 3 of 19, and practical site-level details such as photos and access notes reported by 4 of 19, of whom three were domain experts and one was a technical evaluator.

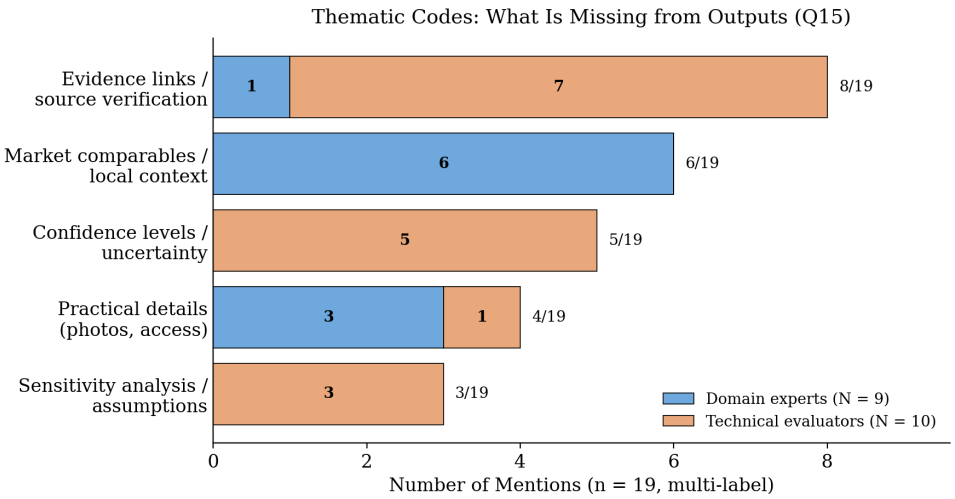


Figure 27: Thematic codes from Q15 (missing information), split by evaluator group ($N = 19$). Evidence provenance and confidence estimates concentrate among technical evaluators, while practical site-level details concentrate among domain experts.

All 19 participants addressed trust in their responses, and the dominant pattern (14/19, 74%) was *conditional trust*, namely willingness to use the outputs as internal first drafts but not as client-facing deliverables without human review. P6[T] described them as *"strong first drafts for an internal workflow,"* P5[D] would *"trust it as a first internal draft, but not something I'd send directly to a client,"* and P18[D] noted they are *"more complete than a rushed manual summary, but I'd definitely edit them."* Four participants (21%), all from the domain group, expressed stronger reservations: P8[D] was *"a bit hesitant to trust it in a real high-stakes case,"* and P16[D] stated reliance *"mainly on my own review."* No participant expressed unconditional trust. This aligns with the framework's human-in-the-loop design (Section 4.7) and confirms the system is perceived as decision support rather than an autonomous decision-maker.

A recurring concern among domain experts was the technical density of outputs. Six domain-expert responses (67% of the domain group) identified readability as a barrier. P16[D] noted that *"the outputs need more plain business language and less system wording,"* P12[D] was concerned that *"some business people may not read all of it,"* and P9[D] found parts of the decision trace *"more technical than necessary."* This is complementary to the transparency results in Subsection 6.3.4. The mechanisms are structurally effective but require a presentation layer adapted to non-technical audiences. Suggestions included shorter executive summaries (P13[D]), clearer separation between verified facts

and assumptions (P11[D], P18[D]), and reduced jargon (P16[D]).

Six participants (32%), all domain experts, identified a gap in contextual business intelligence. Requested additions included local market rent comparables (P2[D], P5[D]), negotiation leverage (P2[D], P9[D]), road access and labour availability (P12[D], P13[D]), and competitor activity (P13[D]). P12[D] summarised this as needing *"transport access details, local labor availability, and whether the landlord is actually responsive."* This reflects a limitation of the current prototype, which operates on a structured warehouse database without external market data feeds. The framework's architecture supports such enrichment through additional specialised agents, but this capability was not implemented.

Among technical evaluators, five of ten explicitly questioned the practical value of the pheromone-based coordination mechanism. P17[T] identified it as *"the least convincing part,"* P6[T] noted it *"may be hard to defend to non-technical teams,"* and P15[T] suggested it *"may be hard to operationalise."* P10[T] characterised it as *"a bit opaque."* This is consistent with the moderate quantitative rating reported in Table 34 and suggests the bio-inspired metaphor may benefit from stronger empirical validation. It is worth noting that some of these concerns may reflect limited exposure to the implementation details rather than inherent shortcomings of the mechanism. The pheromone layer is operational within the prototype, drives concrete bid-scoring adjustments across runs, and is empirically evaluated in the convergence analysis of Subsection 6.2.2. Critiques framed around operationalisability or opacity therefore appear in part to stem from the inherent difficulty of conveying a stigmergic coordination mechanism through a short survey artefact rather than from limitations of the mechanism as built.

Finally, a minority theme (4 participants, all technical) identified the system's transparent failure handling as a distinguishing strength. P15[T] stated that *"the failure case actually increased my trust because the system exposed the problem instead of hiding it,"* and P3[T] ranked Output B above Output A *"because at least the failure is visible and traceable."* This provides qualitative support for the multi-tier failure handling design (Section 4.9). Even when the system cannot solve a problem, making the failure visible and structured is perceived as more trustworthy than silently producing degraded outputs.

6.3.8 Statistical Comparison and Cross-Cutting Observations

Table 36 consolidates the Mann–Whitney U test results for all 15 items rated by both evaluator groups. Of the 15 comparisons, 9 yielded statistically significant differences at $p < .05$, all in the same direction, with technical evaluators rating the system more favourably than domain experts. Effect sizes ranged from small ($r = 0.13$) to large ($r = 0.67$).

The non-significant items are equally informative. Warehouse matching accuracy (both scenarios), compliance checking (GreenLeaf), and novelty of information (both scenarios) showed no significant group difference, indicating that both groups agreed on the system's factual and regulatory accuracy. These items represent the core reasoning outputs of the Multi-Agent pipeline, that is, the parts driven by the ICNP bidding, SHACL validation, and LLM-based recommendation logic.

The divergence concentrates on two specific areas. First, *presentation* items (proposal quality in both scenarios, fallback recommendation usefulness) showed significant differences with medium to large effect sizes ($r = 0.41$–0.67). These items assess the system's ability to produce polished, client-ready deliverables. This is a dimension where domain experts, who routinely prepare such documents, hold higher standards. Second, all four *transparency* items differed significantly ($p < .01$, $r \geq 0.56$), suggesting that the decision

Item	M_D	M_T	U	p	r
Section 1: Output Quality					
Warehouse match (FreshCo)	4.33	4.60	36.0	.432	0.17
Compliance check (FreshCo)	**4.00**	**4.60**	**18.0**	**.007****	**0.51**
Proposal quality (FreshCo)	**3.22**	**4.10**	**9.0**	**<.001*****	**0.67**
Novel info (FreshCo)	3.44	3.80	31.5	.230	0.25
Warehouse match (GreenLeaf)	3.44	3.90	27.0	.101	0.34
Compliance check (GreenLeaf)	4.00	4.20	38.0	.517	0.13
Proposal quality (GreenLeaf)	**3.11**	**3.60**	**23.0**	**.036***	**0.41**
Novel info (GreenLeaf)	3.33	3.80	27.0	.108	0.34
Failure handled reasonably	3.22	3.90	24.0	.070	0.39
Fallback recommendation	**2.44**	**3.20**	**19.5**	**.028***	**0.48**
Section 2: Decision Transparency					
Understand why selected	**3.78**	**4.60**	**14.0**	**.005****	**0.58**
Bidding scores transparent	**3.67**	**4.60**	**12.0**	**.003****	**0.62**
Audit trail sufficient	**3.67**	**4.60**	**15.0**	**.010****	**0.56**
Can identify error location	**3.22**	**4.30**	**10.5**	**.002****	**0.65**
Overall					
Overall usefulness	**3.44**	**4.20**	**18.5**	**.019***	**0.50**

Table 36: Mann–Whitney U tests comparing domain experts ($n = 9$) and technical evaluators ($n = 10$) across all 15 items rated by both groups. Effect size $r = |Z|/\sqrt{N}$. Significance markers are $^{*}p < .05$, $^{**}p < .01$, and $^{***}p < .001$. Statistically significant rows are bolded. The pattern of significant rows concentrates on presentation and transparency items, while items measuring factual accuracy show no group difference.

trace format is well-suited for technical auditing but insufficiently accessible for business stakeholders.

This pattern supports a clear conclusion. The framework's reasoning pipeline produces accurate and useful results that both groups recognise, but its communication layer, the proposal generation and transparency presentation, requires adaptation for domain practitioners. This finding has direct implications for future development. The core Multi-Agent architecture need not change, but the last-mile output formatting should offer configurable verbosity levels and audience-appropriate summaries.

The divergence follows a consistent strength hierarchy across the evaluation dimensions. The system's outputs were rated highest on factual dimensions (matching accuracy, compliance correctness), followed by structural transparency (bidding traces, audit trails), and lowest on presentation quality (proposal professionalism, fallback actionability). This ordering suggests a clear maturity path for the prototype. The reasoning pipeline is effective, but the last-mile delivery to users requires refinement.

Despite these group-level differences, both evaluator groups independently converged on the same trust model. The system is suitable for internal first-draft generation with mandatory human review before external use. No participant endorsed fully autonomous use, regardless of expertise or AI familiarity. This consistent positioning validates the framework's human-in-the-loop design (Section 4.7) and suggests that the approval routing mechanism is not merely a safety feature but reflects genuine user expectations for AI-assisted decision support in regulated domains.

6.4 Summary

This chapter has presented the empirical evaluation of the proposed framework through a 790-run quantitative programme spanning ablation, convergence, priority-routing, validation-layer interaction, and failure-recovery experiments, complemented by a qualitative user perception survey with 19 participants. The evidence gathered across these complementary studies allows each of the five hypotheses introduced at the start of the chapter to be assessed against measured outcomes, with the consolidated findings reported in Table 37.

Hypothesis	Prediction	Key Evidence	Primary Statistic	Verdict
H1 Stigmergic Learning	C0 converges in < 15 runs, C1 remains flat	C0: 10%→96% ($\Delta = +86$ pp), converging at run 14, C1: 60%→56% ($\Delta = -4$ pp)	n/a	**Supported**
H2 Competitive Selection	C2 produces the largest drop	C0 = 0.98 vs. C2 = 0.42 ($\Delta = +56$ pp), C2 matches random allocation	$\delta = 0.774$	**Supported**
H3 Super-additive Safety	Both layers jointly reduce error propagation more than either alone	Factorial IE = +21 pp (super-additive), zero detection overlap (6/6 structural, 3/3 domain), TPR = 1.000, F1 = 0.992	IE = +21 pp	**Supported**
H4 Adaptive Replanning	Multi-tier recovery achieves reliable failure recovery, and the Strategist improves estimator accuracy	Recovery rate: C0 = 100%, C2 = 30%, C7 = 0%, estimator ACC: 96% vs. 90% ($\Delta = +6$ pp), overall $\delta = 0.060$	$\delta = 0.060$	**Supported**
H5 Framework Justification	C0 outperforms C7 at bounded cost	ACC +48 pp ($\delta = 0.576$); VR +40 pp; latency +6.6%; tokens +9.7%	$\delta = 0.576$	**Supported**

Table 37: Hypothesis evaluation summary.

All five hypotheses are supported. The results establish a clear contribution hierarchy among the framework's coordination mechanisms.

1. **ICNP bidding**, with Cliff's δ of 0.774, is the indispensable core mechanism. Without it, the system reduces to random allocation regardless of agent quality.
2. **Pheromone adaptation**, with a convergence Δ of +86 pp, is essential for long-term learning and context-sensitive routing, though invisible in single-run evaluation.
3. **Validation layers**, with IE of +21 pp, TPR of 1.000, and zero-overlap complementarity, are critical for safety assurance with a super-additive interaction on accuracy. Each layer alone degrades ACC, but the combination restores it while providing complete coverage across all 9 of 9 injected defects with zero detection overlap.
4. **LLM Strategist**, with Cliff's δ of 0.060 for overall accuracy, estimator ACC of +6 pp, and 100% failure recovery, provides the Tier 2 replanning capability of the self-healing stack. Its latency overhead of +48 s is the cost of adaptive resilience.
5. **Ethics Checker**, with Cliff's δ of 0.121, contributes a statistically negligible accuracy improvement in isolation, but it provides indispensable governance screening in any compliance-sensitive deployment context.

This hierarchy informs deployment decisions. A minimal viable configuration requires the ICNP and pheromone engine for core coordination. Production deployments should include the LLM Strategist for adaptive failure recovery and all validation layers for safety and governance. The implications of this hierarchy for deployment strategy are developed further in Chapter 8.

All results derive from a single synthetic domain with five client archetypes and ten agents. Although the framework mechanisms are domain-agnostic by design, the specific accuracy values, convergence rates, and violation rates reported here are properties of this evaluation instantiation rather than universal constants. Domains with different characteristics, such as a larger competing agent pool, non-binary ground truths, continuous rather than categorical capability matching, or violation rates above 40%, may produce different quantitative outcomes while leaving the qualitative ordering of mechanisms unchanged. The most uncertain extrapolation concerns pheromone dynamics at scale. With three warehouse agents, each run provides a strong and unambiguous reinforcement signal. With 50 competing agents, pheromone mass would be more diffuse, potentially slowing convergence or requiring decay parameter recalibration. These scale effects remain open research questions that a follow-up study with a larger synthetic population could address.

In the next chapter, the discussion broadens beyond the empirical findings to consider the sustainability and ethical implications of the framework, examining how its design decisions interact with emerging regulatory expectations and what they imply for responsible deployment in practice.

7. Sustainability Analysis and Ethical Implications

This chapter analyses the proposed framework from the perspective of sustainability and ethics, following the structure recommended by the UPC guide for TFG/TFM sustainability reports [100]. The analysis is organised along the three dimensions of the sustainability matrix, namely environmental, economic, and social, and within each dimension it addresses the development of the thesis, the execution of the deployed project, and the associated risks and limitations. Because this work is primarily a research prototype that has been partially deployed and evaluated, some parts of the analysis are grounded in the actual implementation developed during this thesis, while others are prospective, referring to the expected impact of a full production deployment.

7.1 Environmental Dimension

7.1.1 Development of the Thesis

The environmental footprint of the thesis development is primarily electricity consumption from programming, experimentation, model inference, and document preparation. Unlike projects involving physical prototyping or hardware fabrication, material consumption is negligible, since the work is almost entirely digital. The primary resources consumed during development were a personal laptop, cloud services for containerised testing, and external AI inference services for the experimental evaluation.

The laptop used throughout the project is estimated to have consumed approximately 0.05 kWh per hour of active use. Assuming approximately twenty weeks of development at twenty-five active hours per week, this yields an estimated total of 500 active hours and a direct electricity consumption of approximately 25 kWh from the personal computer alone. At the average carbon intensity of the Spanish electricity grid in 2024–25, which is approximately 0.18 kg CO_2e per kWh [101], this corresponds to roughly 4.5 kg CO_2e from personal computing. The OpenAI API was used to run all experimental evaluations at a total monetary cost of 10 EUR, corresponding to a small and bounded number of inference calls. While the exact energy cost of those calls is not publicly disclosed by the provider, the selective nature of the experiments kept this contribution modest.

Several design decisions were taken specifically to limit the environmental impact of the development phase. First, the system is implemented as a modular and containerised architecture, which enables reproducibility and environment reuse without duplicating hardware setups. Second, experiments were bounded and selective rather than exhaustive, meaning that unnecessary inference runs were deliberately avoided. Third, expensive large language model (LLM) calls were treated as a scarce resource throughout both design and testing, and the framework was instrumented to favour cached outputs and smaller models wherever possible. These decisions collectively reduce computation time, energy consumption, and unnecessary data storage while preserving the validity of the technical evaluation [102].

7.1.2 Project Execution

When the framework is deployed in a real operational scenario, its environmental impact depends directly on the scale of usage, the hosting infrastructure, and the frequency and volume of model inference requests. The architecture is designed to minimise unnecessary

consumption through on-demand execution. Agents run inside containers only when a task requires them and scale down to zero when idle, which avoids keeping all system capabilities active at all times. This is especially important in workloads characterised by usage peaks and prolonged idle periods, where always-on deployments would waste significant energy.

The framework also includes mechanisms that support a more sustainable operational lifecycle. Stateless services can scale horizontally behind a load balancer, and stateful services use retention strategies such as time-to-live policies for markers, intermediate results, and completed task metadata [103]. This limits long-term storage growth and prevents the accumulation of unnecessary artefacts. Older execution records can be archived and removed from the live system, which is consistent with the principle of retaining only data that is operationally necessary.

From a circular economy perspective, the project favours software reuse and modularity. The same infrastructure can be adapted to different coordination scenarios without redesigning the full system, and individual components such as the orchestrator, the marker model, and the audit layer can be reused or extended in future work. The principal environmental cost is therefore not material waste but computational and storage consumption throughout the operational lifecycle of the system.

7.1.3 Environmental Risks and Limitations

The main environmental risk is that the actual footprint may become significantly larger than anticipated if usage scales substantially, if large models are invoked too frequently, or if the system is deployed on infrastructure powered by a carbon-intensive energy mix. A second risk is the rebound effect, whereby efficiency improvements lower the perceived cost of use and thereby encourage greater aggregate consumption. There is also inherent uncertainty regarding the environmental footprint of external AI providers, since energy mix, cooling system efficiency, and hardware utilisation rates are not always transparently reported.

The principal limitation of this environmental analysis is that no full lifecycle assessment or direct carbon accounting has been carried out. The environmental evaluation is therefore qualitative and design-oriented rather than a precisely quantified footprint. A future production deployment should complement this section with operational metrics including energy consumption per inference call, total model invocation counts, storage growth rates, and provider-specific emissions factors.

7.2 Economic Dimension

7.2.1 Development of the Thesis

The primary cost of the thesis development is the author's time, which encompasses literature review, system design, implementation, experimentation, debugging, and academic writing. The project was carried out during 2025 over approximately twenty weeks at an average of twenty-five hours per week, yielding an estimated total of 500 hours of work. Applying a reference hourly rate of 25 EUR per hour, consistent with junior engineering labour costs in the Catalan market in 2025, the personnel cost of the development is estimated at 12,500 EUR. This figure represents the dominant cost item and would constitute the main line of an invoice to a hypothetical client.

Direct material costs are negligible because the project does not require physical components. The only significant direct expenditure beyond personnel time was the use of

the OpenAI API for running the experimental evaluation, which amounted to a total of 10 EUR. This low cost reflects the bounded and selective approach to experimentation described in Subsection 7.1.1. Most other software resources used during development are open-source or already available in standard academic and engineering environments, and no additional cloud infrastructure costs were incurred during the development phase, since all containerised testing was performed locally. Table 38 summarises the estimated development cost breakdown.

Item	**Quantity**	**Cost (EUR)**
Personnel time	500 h @ 25 EUR/h	12,500
OpenAI API inference	10 EUR flat	10
Hardware and peripherals	Personal laptop (amortised)	∼0
Software licences	Open-source only	0
Cloud infrastructure	Local testing only	0
Total		**12,510**

Table 38: Estimated development cost breakdown.

Cost-optimisation decisions made during the project include the reuse of existing open-source frameworks and libraries, the adoption of a modular architecture that reduces the need for redundant implementation, and the systematic limitation of expensive LLM interactions through caching, bounded invocation counts, and the use of simpler rule-based components wherever appropriate.

7.2.2 Project Execution

From a viability standpoint, the proposed framework is economically feasible when the benefits of automation, traceability, and scalability outweigh the operating costs of infrastructure and model inference. Its microservice architecture supports gradual deployment, meaning that the system does not require a large upfront investment in dedicated hardware. Instead, it can be rolled out incrementally on cloud-native infrastructure and scaled in proportion to demand [52, 104].

The main recurring execution costs in a production deployment would be associated with compute resources, persistent storage, message brokering, monitoring, maintenance, and model inference calls. Among these, LLM inference is likely to be the most variable and cost-sensitive item, since it depends on request volume, context window size, provider pricing, and the frequency with which the LLM Strategist is invoked. Maintenance costs would include software updates, security patches, prompt and policy revisions, monitoring operations, and the adaptation of rules or ontologies to new application domains.

At the end of the system lifecycle, decommissioning costs are expected to be moderate because the architecture is modular and based on standard technologies. Containers, infrastructure-as-code definitions, and reusable software components facilitate migration, component replacement, or controlled shutdown. Many of the project artefacts are also reusable beyond this thesis, including the orchestration approach, the audit model, and the design principles for safe integration of LLM-based strategists into Multi-Agent workflows.

7.2.3 Economic Risks and Limitations

The principal economic risk is cost uncertainty, particularly with regard to external AI services. Pricing changes by providers, vendor lock-in, or unexpected growth in usage volume could reduce the cost-effectiveness of the solution. A related risk is that operating

costs may be systematically underestimated if large volumes of markers, execution logs, or orchestration events must be retained for extended periods. There is also a feasibility risk arising from integration complexity, in that the framework may be technically sound but economically unjustifiable in domains where a simpler workflow engine would be sufficient.

The primary limitation of this economic analysis is that it is not grounded in a full production deployment with real traffic and real operational budgets. The discussion therefore focuses on cost structure and feasibility reasoning rather than on a precise financial forecast. A future industrial evaluation should include concrete workload assumptions, current provider pricing schedules, maintenance cost plans, and a comparative analysis against alternative orchestration solutions.

7.3 Social Dimension

7.3.1 Development of the Thesis

At the level of thesis development, the work has had a positive personal and educational impact. It has contributed to the author's formation in distributed systems, knowledge representation, software architecture, and responsible artificial intelligence. It has also fostered awareness of the social consequences of automation, including the importance of accountability, traceability, and human oversight in systems that coordinate autonomous agents.

In accordance with good professional and academic practice, this work has been produced using inclusive and non-sexist language, and it avoids presenting artificial agents as autonomous moral subjects in ways that would obscure human responsibility. This consideration is particularly important in academic writing on AI, where anthropomorphic language can generate misleading expectations about system capabilities and the allocation of ethical accountability.

The work complies with the UPC Code of Ethics [105] and the applicable research integrity guidelines, as formalised at the time of thesis enrolment.

7.3.2 Project Execution

Because the framework has been deployed and actively evaluated, its social impact is not merely potential but already relevant in practice. The system addresses a real need for more transparent, scalable, and auditable coordination of Multi-Agent processes, particularly in contexts where operators need to understand why decisions were taken and how actions were triggered. The audit trail, the provenance model, and the structured task allocation mechanisms improve accountability and help reduce the opacity that is commonly associated with complex AI-based systems.

The primary beneficiaries of such a system include software developers, system operators, organisations deploying complex automated workflows, and end users who depend on reliable and traceable service execution. At the same time, the system may also produce negative social effects if it is used irresponsibly or governed inadequately. A highly automated orchestration framework may encourage over-reliance on machine-generated recommendations, reduce meaningful human supervision, or be deployed in organisational settings to intensify monitoring and control of workers. If task-allocation criteria are not carefully designed, certain actors, whether software agents, organisational teams, or human participants, may be systematically disadvantaged through repeated exclusion or unfair workload concentration [106, 107].

The potential for bias is also a relevant concern in the deployed system. The LLM Strategist may produce suggestions that reflect biased assumptions or ethically problematic shortcuts embedded in training data. For this reason, the framework does not treat LLM output as unconditionally valid. Suggestions are filtered through validation rules, ethics checks, and, where necessary, human approval. This reduces the risk that unsafe, discriminatory, or privacy-invasive proposals are executed directly. Accessibility and usability are additional socially important factors. For the framework to support responsible use, the information it produces must be understandable to non-expert stakeholders. Human-readable labels, structured logs, and explicit reasons for task assignment improve interpretability. The system has been evaluated through a user survey, which provides initial empirical evidence regarding usability and user perception.

7.3.3 Social Risks and Limitations

The main social risks are misuse, over-automation, practical opacity despite formal traceability, and unfair or biased decision-making. A further relevant risk concerns data privacy. Because the system may store user identifiers, task histories, or behavioural traces in operational logs, inappropriate data retention or insufficient access control could adversely affect data subjects. In the current deployment, this risk is mitigated through data retention policies that limit how long operational data and logs are kept, together with controlled access to system information.

There is also a dependency risk, whereby an organisation may become overly reliant on a complex orchestration framework that is difficult to govern without specialised technical knowledge. The principal limitation of the social analysis is not the absence of real-world use, since the framework has been deployed and evaluated through a user survey, but rather the limited scope of that evaluation. Although the survey provides empirical evidence on user perception and usability, the analysis remains constrained by the size and context of the study conducted within this thesis. Broader longitudinal and participatory evaluation would therefore be needed to assess long-term organisational and social effects more comprehensively.

7.4 Ethical Implications

This work addresses the need for transparent, safe, and adaptable coordination of complex agent-based systems. These needs are defined by both the technical literature and the practical requirement for accountability in environments where automated decisions may affect services, workflows, and human stakeholders. The central ethical challenge is therefore not only whether the framework functions correctly, but whether it can be operated in a manner that respects safety, fairness, privacy, and human oversight.

A first ethical issue concerns unintended consequences. A system designed to optimise task coordination could equally be used to accelerate harmful objectives, to bypass human judgement, or to normalise excessive dependence on machine-generated plans. Even when the LLM Strategist plays only an advisory role, its suggestions may influence decisions in ways that appear locally efficient but conflict with broader ethical or legal constraints. For this reason, the architecture incorporates several safeguards. The LLM does not directly execute any actions, proposals are validated against graph constraints and runtime rules, and high-risk actions require explicit human approval before execution.

A second issue concerns fairness and the principle of non-maleficence. The framework must not produce avoidable harm to users, workers, or affected third parties. In particular, the allocation mechanism must not systematically advantage certain actors without

justification, and privacy-related or safety-critical tasks must not be removed or deprioritised solely because doing so would improve short-term efficiency metrics. The inclusion of an Ethics-Check Agent and explicit validation constraints is intended to provide a formal barrier against such behaviour, although ethical governance ultimately also depends on the policies defined and maintained by system administrators.

Data protection is a further important consideration. Because the system stores operational logs that may contain identifiers or task-related traces, these data are managed under controlled retention policies and are not publicly disclosed or shared externally. Access is restricted to what is strictly necessary for operational and evaluation purposes. The user survey conducted as part of this thesis followed applicable GDPR guidelines [3] for the collection and processing of participant data. More generally, the project adheres to the principles of data minimisation, purpose limitation, controlled retention, and confidentiality in the handling of both operational and evaluation data.

Taken together, these three considerations (the risk of unintended consequences, the requirement for fairness and non-maleficence, and the obligation to protect personal data) define the ethical boundaries within which the proposed framework must be developed and operated.

The overarching ethical position of this thesis is that greater automation must be accompanied by proportionally greater accountability. The value of the proposed framework lies not only in its orchestration capability but also in its potential to make decisions more inspectable, contestable, and governable than would be possible with purely opaque alternatives.

7.5 Relation to the Sustainable Development Goals

This work contributes indirectly to several Sustainable Development Goals (SDGs) [12], although its contribution is primarily technological and enabling rather than direct social or environmental intervention.

The work relates most directly to **SDG 9** (Industry, Innovation and Infrastructure) because it proposes a scalable and modular architecture for coordinating intelligent services in complex digital environments. The framework supports innovation in software infrastructure through interoperability, observability, and safe integration of AI-based components. By providing a reusable and extensible foundation for Multi-Agent coordination, it contributes to more resilient and intelligent digital infrastructure.

It also relates to **SDG 12** (Responsible Consumption and Production), because the design explicitly seeks to reduce unnecessary computational consumption through on-demand execution, bounded LLM usage, caching strategies, and retention policies for temporary artefacts. While these measures do not eliminate environmental cost, they promote a more responsible and deliberate use of digital resources throughout the system lifecycle.

Finally, it relates to **SDG 16** (Peace, Justice and Strong Institutions), particularly through its emphasis on traceability, explainability, and accountability in automated decision-making. Systems that support auditability and human oversight are better aligned with the principles of responsible governance than systems that make opaque decisions without clear provenance, human control, or mechanisms for contestation.

These contributions are necessarily bounded by the scope of the thesis. The project does not directly address a social or environmental problem at population scale, but it does propose design principles that can inform the development of more responsible digital infrastructures across a range of application domains.

8. Conclusions, Limitations and Future Work

This chapter closes the thesis by reflecting on what the design and empirical work presented in the preceding chapters has established, where its limits lie, and which directions remain open for further investigation. The conclusions revisit the research question and summarise the principal findings, the limitations section qualifies their interpretation along internal, external, and construct dimensions, and the future work section traces a research agenda component by component, mapping each line of work to the mechanism it would strengthen. The chapter ends with a closing reflection on the broader contribution of the framework to the design of adaptive, auditable, and governance-aware Multi-Agent orchestration.

8.1 Conclusions

This thesis investigated whether fusing swarm-inspired coordination, a formal ontological substrate, and layered governance into an LLM-based Multi-Agent framework can close three architectural gaps left unaddressed by existing orchestration platforms. These gaps are stateless agent-performance tracking, the absence of a machine-interpretable capability model, and the lack of structural compliance enforcement. The work pursued this question through a unified design and empirical programme. On the design side, Chapter 4 proposed an integrated architecture that fuses stigmergic pheromone markers, an Improved Contract Net Protocol with pheromone-weighted bidding, an OWL/RDF ontological substrate, a SHACL structural validator, an Ethics-Check Gate, and a multi-tier failure-recovery stack. Chapter 5 instantiated this architecture as a cloud-native prototype of event-driven microservices. On the empirical side, Chapter 6 evaluated the prototype across 790 controlled pipeline runs in a commercial warehouse brokerage domain, covering an ablation study of nine framework configurations, a 300-run pheromone convergence experiment, a validation-layer interaction analysis, a failure-injection recovery study, a priority-based estimator-selection analysis, and a user perception survey with $N = 19$ domain and technical evaluators.

The quantitative evidence supports all five hypotheses stated in Chapter 6. Stigmergic learning drives assignment accuracy from $\sim$10% to 96% within 14 convergence runs (H1). Competitive selection through ICNP bidding proves to be the single most consequential mechanism, as replacing it with round-robin allocation degrades accuracy to 42% and reproduces the behaviour of random assignment (H2). The two validation layers interact super-additively on downstream accuracy with IE = +21 pp, and jointly achieve perfect violation recall (TPR = 1.000, FPR = 0.011, F1 = 0.992) with zero detection overlap between the structural and domain classes (H3). Multi-tier recovery achieves 100% successful recovery under the full framework against 30% for round-robin allocation and 0% for the uncoordinated baseline (H4). The full framework outperforms the coordination-free baseline by 48 percentage points in assignment accuracy and 40 percentage points in violation detection, at a measured cost of only 6.6% additional latency and 9.7% additional token consumption (H5). The user survey corroborates the quantitative findings, with 84% of participants selecting the full framework output as the most trusted and 95% as the most suitable for audit, while converging on a consistent trust model in which the system is positioned as a decision-support tool requiring mandatory human review before external use.

Taken together, these findings establish a clear ordering of the marginal contributions of the coordination mechanisms to overall system quality. The ICNP bidding layer (Cliff's $\delta = 0.774$) constitutes the indispensable core without which the system reduces to random allocation regardless of agent quality. Pheromone adaptation ($\Delta = +86$ pp over the convergence horizon) is essential for long-term learning and context-sensitive routing, though its contribution is invisible to single-run evaluation. The validation layers (IE $= +21$ pp, TPR $= 1.000$) are critical for safety assurance, with each layer alone degrading accuracy but the combination restoring it while covering all nine injected defect classes. The LLM Strategist enables the Tier 2 replanning capability of the self-healing stack and contributes a +6 pp improvement in estimator-selection accuracy. The Ethics Checker, although statistically negligible in isolation ($\delta = 0.121$), provides indispensable governance screening in any compliance-sensitive deployment.

The overall answer to the research question is affirmative. The three gaps identified in Chapter 1 can be closed by a single architecture. (1) Stateless coordination is addressed by treating coordination as environmental signalling over a shared knowledge graph. (2) The absence of a formal capability model is addressed by ontological capability matching. (3) The lack of domain-specific governance is addressed by a stratified validation pipeline. The same architecture delivers empirical gains at production-realistic overhead, while remaining domain-agnostic in its mechanisms and pluggable in its agents and constraints.

8.2 Limitations

Despite the consistency of the results, a number of limitations temper their interpretation. Some concern the internal integrity of the evaluation, others its external generalisability, and others the constructs used to measure system behaviour.

On the internal side, LLM outputs exhibit residual variance even when temperature is set to zero, because batched inference and floating-point rounding introduce non-determinism that cannot be fully eliminated. Ten replications per scenario per configuration were used throughout. Assignment accuracy is reported as the mean over replications (with IQR), latency and token metrics as medians. The ground-truth assignments were defined by a logistics domain expert from the seed database, with the risk of subjective bias further reduced by the fact that assignments follow directly from client requirements such as cold storage, regional affinity, and budget, all of which are deterministic properties of the data. Finally, the Wilcoxon signed-rank tests conducted over five paired client observations are underpowered and report uniformly non-significant p-values despite meaningful effect sizes. Cliff's delta computed over all 50 individual runs per configuration is therefore the primary measure of practical significance and the basis for the reported contribution hierarchy.

On the external side, all client scenarios are constructed from a synthetic seed dataset rather than live operational data, meaning that the reported accuracy values, convergence rates, and violation rates are properties of this particular evaluation instantiation rather than universal constants. Generalisation to other domains such as healthcare, manufacturing, or human resources requires domain-specific agent re-implementation and SHACL rule adaptation, even though the framework mechanisms are domain-agnostic by construction. The evaluation also uses a small agent pool of three competing warehouse agents and two competing estimator agents, whereas real deployments may involve tens or hundreds of competing agents. The most uncertain extrapolation concerns pheromone dynamics at scale, because with a larger agent population the pheromone mass would be more diffuse across markers, potentially slowing convergence or requiring recalibration

of the decay parameters. All LLM-based agents additionally use a single provider and model configuration, OpenAI GPT-4o-mini (`gpt-4o-mini`, see Table 16), and results may vary with alternative models that differ in instruction-following quality, structured-output reliability, or tokenisation behaviour.

Finally, the user perception survey engaged 19 participants recruited through purposive sampling, which is sufficient to surface consistent cross-group themes such as conditional trust and readability concerns but does not support population-level generalisation. The thematic coding of open-ended responses was carried out by the author alone without a second independent coder, so no inter-rater reliability statistic is reported. The analysis is therefore best read as an interpretive synthesis of recurring participant concerns rather than as a quantitatively validated coding exercise, and the verbatim quotations included in Section 6.3 are provided so that readers can assess the grounding of each theme directly. Longitudinal evaluation with working practitioners in an operational setting, accompanied by multi-coder analysis of the qualitative data, would be required to confirm that the reported patterns persist over sustained use.

8.3 Future Work

The limitations described above suggest a natural research agenda that extends both the architecture and its empirical grounding. The directions below trace the framework component by component rather than forming a disconnected list, so that each line of work can be connected to the mechanism it would strengthen.

The most structurally significant extension concerns agent generation and registration. The current prototype requires each agent to be implemented manually, with its capabilities, tools, and system prompt encoded by a developer before being seeded into the Consul-backed registry. Closing this loop would allow the framework itself to generate and register new agents on demand. In such an extended design, the user would describe a desired capability in natural language, and an LLM-assisted coding loop would synthesise the agent skeleton, propose an initial tool set, produce a system prompt, derive the appropriate capability tokens, and complete the registration without human intervention. Recurring gaps in bid evaluation sequences, where no registered agent meets the capability requirements of incoming tasks, could themselves serve as the signal for which new agents are required, with the framework triggering generation and registration whenever such gaps cross a configurable threshold. The LLM would also inspect the current agent roster and automatically expose tools and capabilities rather than relying on manual declaration, transforming the framework from a fixed-population system into a self-expanding one in which coordination quality and agent specialisation co-evolve through the pheromone engine.

A related architectural extension concerns how agents query domain data. The current warehouse brokerage agents rely on hand-written SQL against a fixed relational schema. A more general approach would expose the database schema to the LLM at inference time and delegate query construction to the model, so that the agent receives the schema description alongside the natural language requirement, produces the corresponding query, executes it, and interprets the result. This would remove the need to rewrite queries whenever the schema evolves and allow the same agent implementation to generalise across different data back-ends. It also introduces a research question for the pheromone engine: whether reinforcement should operate on the agent, on the generated query, or on a joint agent-query pair, and how decay parameters should be calibrated when the space of possible queries becomes effectively unbounded.

Scalability of pheromone dynamics and token overhead both remain empirically open. The current evaluation uses three competing warehouse agents and two estimator agents, a pool small enough that each run produces a strong and unambiguous reinforcement signal. With a larger competing population the pheromone mass would be more diffuse across markers, potentially requiring recalibration of the decay rate λ and the reinforcement step. A controlled study varying the agent pool across 10, 25, and 50 synthetic agents per role would characterise the sensitivity of the convergence trajectory to population size. On the cost side, the measured token overhead of 9.7% relative to the coordination-free baseline is driven by the number of competing agents and the depth of the bidding protocol rather than by pipeline complexity, which makes the overhead predictable in principle as the agent population grows. Empirical confirmation across variable populations is reserved for future work.

The Ethics-Check Agent itself could be extended. The current implementation reasons purely from a fixed system prompt and the plan context provided at review time, so each evaluation begins from the same baseline regardless of what the agent has previously approved or vetoed. Connecting the agent to a persistent knowledge base would close this loop, with past violations, the rationales attached to vetoes, and the corrective actions taken by human reviewers retained as structured records that are retrieved and supplied as context for subsequent reviews. Over time the agent would accumulate a domain-specific case history that informs its judgement on ambiguous plans, reducing the variance of its decisions, surfacing recurring failure patterns to human reviewers, and allowing the framework to demonstrate not only that ethical review took place but that lessons from earlier reviews were carried forward into later ones.

Observability also warrants dedicated investment. The current prototype surfaces runtime state through OpenTelemetry traces, structured audit logs, and SPARQL queries, which are sufficient for engineers but do not give non-technical stakeholders a direct operational view. A monitoring and tracing interface would render pheromone intensities, bid scores, validation verdicts, ethics decisions, escalations, and run-level provenance in a single navigable surface, with the ability to drill from a high-level run summary down to individual agent messages. Such an interface would also serve as a governance artefact by making the audit trail inspectable without SPARQL expertise, directly addressing the accessibility gap that domain experts identified in the user survey.

Output formatting is a further area in need of attention. The qualitative analysis revealed that both evaluator groups rated the factual and reasoning components of the pipeline comparably, while presentation items such as proposal quality and fallback actionability produced statistically significant gaps with medium-to-large effect sizes. The core Multi-Agent architecture need not change to address this, but the last-mile output layer should offer configurable verbosity levels and audience-appropriate summaries. An executive-summary mode for business stakeholders, a clear separation between verified facts and model-inferred assumptions, reduced system-level jargon in client-facing proposals, and actionable fallback recommendations when the primary recovery path fails would all reduce the editing burden that domain experts currently accept before forwarding outputs. Integration of external market-intelligence agents would further extend the system from a structured-data reasoner toward a genuinely context-aware decision-support tool.

Finally, three complementary lines of broader empirical validation remain open. Expanding the client population to 15 to 20 scenarios would restore adequate statistical power to the Wilcoxon signed-rank tests and enable finer-grained per-scenario analyses. Cross-model replication with alternative LLM providers, including open-weight models hosted locally, would separate framework-level effects from model-specific artefacts and test robustness across differences in instruction-following quality and structured-output

reliability. A multi-domain evaluation beyond warehouse brokerage, covering domains such as healthcare triage, legal-document review, or industrial incident handling, would test the portability of the mechanisms claimed to be domain-agnostic and surface any implicit assumptions embedded in the current SHACL shapes and ontology. A production pilot with real operational traffic would additionally provide the longitudinal, cost, and environmental measurements that the sustainability analysis in Chapter 7 could not yet quantify.

8.4 Closing Remarks

The broader claim of this thesis is that adaptive, auditable, and governance-aware Multi-Agent orchestration does not require radically new components, but rather a principled recombination of ideas that have matured independently across three research communities. Stigmergy contributes a decentralised coordination substrate that turns execution history into a first-class environmental signal. The Semantic Web contributes a machine-interpretable vocabulary for capabilities, plans, and provenance, together with a formal validator for structural guarantees. Modern LLM-based orchestration contributes flexible, natural language driven agents capable of reasoning, tool use, and strategic replanning. None of these ingredients is individually sufficient, because stigmergy without semantics lacks auditable ground, semantics without learning lacks adaptivity, and LLM orchestration without either leaves coordination stateless and governance optional. Their integration, as realised in this thesis, produces a framework whose quality grows with use, whose decisions are traceable by construction, and whose safety is stratified across complementary abstraction levels. That this integration can be delivered at bounded latency and token cost, and that the resulting system is perceived by both technical and domain evaluators as a trustworthy first-draft generator requiring human review, suggests that the coordination gap between current orchestration platforms and the accountability demands of regulated AI deployments is not only narrow, but concretely closable with the techniques demonstrated here.

Appendices

A. Pipeline Walkthrough: FreshCo C0 Run

This appendix traces a complete pipeline execution for the FreshCo client through the full framework configuration (C0), to ground the quantitative results of Chapter 6 in a concrete example.

A.1 Client Request

The pipeline receives the following payload P at its intake endpoint, in the natural-language format:

> *"FreshCo, a food-distribution company, requires a cold-storage warehouse in Southern Europe to support its distribution operations. Monthly budget: EUR 18,000. The facility must meet food-grade compliance requirements. This request is high priority."*

Stage 1 of Algorithm 1 extracts the required contextual metadata M from the payload:

- **Client:** FreshCo (food-distribution company)
- **User context:** authorised tenant account
- **Submission channel:** REST intake endpoint
- **Declared priority:** high
- **Declared constraints:** budget EUR 18,000/month, cold storage, food-grade compliance

All required fields are present, so the payload is admitted to Stage 2 (content filtering) rather than rejected. Stage 2 detects no policy violations, and the request proceeds to structuring.

A.2 Task Normalization

In Stage 3 the Supervisor Agent invokes an LLM to emit a `stig:TaskSpec`. For the FreshCo request the TaskSpec instance contains:

- **task_id:** `stig:task/freshco-cold-storage-es-2026`
- **description:** "Locate and contract a cold-storage warehouse in Southern Europe for food distribution, budget EUR 18,000/month, food-grade compliance required."
- **required_capabilities:** {`transcription`, `form_filling`, `warehouse_matching`, `deal_estimation`, `compliance_check`, `proposal_generation`}
- **tags:** {"food-distribution", "cold-storage", "southern-europe", "food-grade", "HACCP"}
- **risk_level:** `high`
- **priority:** `high`
- **requires_human_approval:** `True`

The following orchestration identifiers are attached to the TaskSpec when it is persisted to the graph marker store:

- **Pipeline ID:** `eb595973-d0cf-49b1-8e13-8e92dbf283f9`
- **Run ID:** `14c503ab-a83d-4647-b48d-b13d56d57295`
- **Tasks generated from TaskSpec:** 12
- **Strategy emitted:** True
- **Ethics-check scheduled:** True

A.3 Stigmergic Task Assignment via ICNP

Task assignment follows Algorithm 2. For each derived task the system issues a CFP. Each registered agent computes its stimulus score S (Eq. 2) against its dynamic response threshold θ_i (Eq. 3) and, if admitted, generates a bid score B (Eq. 5). The winning bid produces a `stig:AffordanceMarker`. The two competitive rounds for this run are shown below.

A.3.1 Round 1 – Warehouse Matching (3-way bid)

CFP parameters: required capability `warehouse_matching`, priority `high`, initial stimulus $s_0 = 1.0$.

Agent	Bid score B	Region tag	Outcome
warehouse-south	0.6988	Southern Europe (Milan, Bologna, Rome)	Winner
warehouse-central	0.3387	DACH + France (Munich, Lyon, Paris)	Runner-up
warehouse-north	0.3197	Scandinavia (Gothenburg)	Runner-up

- **Bid delta (winner − runner-up):** 0.3600
- **Expected winner:** `warehouse-south` (FreshCo requests Southern Europe)
- **Actual winner:** `warehouse-south` (correct)

The margin is explained by tag overlap on the `southern-europe` tag, the capability-match term of B (Eq. 5) contributes equally to all three bidders.

A.3.2 Round 2 – Deal Estimation (2-way bid)

CFP parameters: required capability `deal_estimation`, priority `high`. The two competing estimators differ only in their secondary capability (`financial_analysis` vs. `timeline_analysis`) and optimisation strategy.

Agent	Bid score B	Strategy	Outcome
cost-estimator	0.6955	Cost-optimised	Winner
speed-estimator	0.3196	Speed-optimised	Runner-up

- **Expected winner:** `cost-estimator` (FreshCo has high priority but a firm EUR 18,000/month budget; the cost-optimised strategy aligns with the TaskSpec's budget constraint).
- **Actual winner:** `cost-estimator` (correct).

A.4 Plan Generation, SHACL Validation, and DAG Construction

The Graph Builder Agent synthesises the affordance markers from both bidding rounds into an execution graph, then submits it to the validation pipeline.

SHACL validation (Algorithm 3). Both the `pyshacl` reasoner and the programmatic validator approve the plan on the first pass:

- **C1 – Structural completeness:** passed (all 12 task nodes have an assigned agent, a non-empty description, and a non-empty capability set).

- **C2 – Graph integrity:** passed (no dangling edges, Kahn's algorithm confirms acyclicity).
- **C3 – Capability alignment:** passed.
- **C4 – Ontology conformance:** passed.
- **final_conforms:** True.
- **Iterative refinements applied:** 0.

DAG construction (Algorithm 4).
- **Tasks:** 12
- **Maximum topological depth:** 5
- **Parallel phases $\{\Phi_k\}$:** $|\Phi_0| = 1$ (Call Transcription root), $|\Phi_1| = 2$, $|\Phi_2| = 5$ (warehouse + estimator bidding winners, compliance fan-in), $|\Phi_3| = 2$, $|\Phi_4| = 1$ (Proposal Template leaf).
- **Critical path length:** 5 tasks (intake → form fill → warehouse match → compliance → proposal).
- **Fan-out / fan-in nodes:** Deal Summary acts as the primary fan-out, Proposal Template is the terminal fan-in join.

A.5 Human Validation and Approval

Because the TaskSpec carries `risk_level = high` and `requires_human_approval = True`, both triggering conditions at Section 4.7 fire and the plan is routed to the *mandatory human sign-off* workflow rather than directly to user plan validation.

Mandatory sign-off. The approver is presented with the full plan context (task assignments, dependency structure, SHACL report, risk factors) and approves the plan. The structured sign-off record contains:
- **Approver:** pipeline operator
- **Authorisation level:** tenant-admin
- **Decision:** approved
- **Rationale:** "Plan satisfies FreshCo's cold-storage and budget constraints, winning agent holds required food-grade certifications."

User plan validation. After sign-off the plan is presented to the user, who accepts it without modification.

A.6 Execution, Adaptation, and Failure Handling

A.6.1 Ethics-Check Gate

Immediately before execution, the Ethics-Check Agent (Section 4.8) performs its four domain-agnostic checks:
- **No hallucinated data:** passed.
- **No unfair bias:** passed.
- **No sensitive-data leakage:** passed.
- **No unsafe approvals:** passed (mandatory human sign-off present).
- **Veto:** none.

A.6.2 Domain Compliance Check

A domain-specific Compliance agent then audits the plan against FreshCo's business rules. Unlike the domain-agnostic Ethics-Check Agent, the Compliance agent (Table 5) is deterministic and rule-based. It evaluates each candidate warehouse-client pairing against the five rules defined in Chapter 6, cold storage, hazmat certification, security level, square-footage range, and budget compliance, and returns structured pass/fail results scoped to the upstream-suggested pairing when possible.

- **Passed:** False
- **Blocks:** 2 – two warehouses in the inventory (Bavaria Logistics Hub, Centro Logistico Bologna) fail the cold-storage rule and are blocked for any HACCP-tagged TaskSpec.
- **Warnings:** 4
- **Scoped to winning assignment:** True – the selected `warehouse-south` passes all five rules, blocks apply only to alternatives that would have been selected under relaxed constraints.

Because the blocks do not affect the winning assignment, execution proceeds.

A.6.3 Execution Summary

- **Tasks completed:** 12/12
- **Tasks failed:** 0
- **Strategist deltas applied:** 0 (no runtime replanning required).
- **End-to-end latency:** 440.6 s. This run sits in the upper tail of the C0 distribution (median 210 s, IQR 177–239 s). The inflated value reflects first-run cold-start effects (cache warm-up, transient OpenAI API latency) rather than steady-state behaviour.
- **Token consumption:** 31,413 total (20,942 input, 10,471 output).

Because no task fails, none of Tiers 1–3 of Algorithm 6 is entered.

A.6.4 Pheromone State After Run

On successful completion, the Pheromone Engine applies the reinforcement branch of Eq. 6 to the two competitive winners. The resulting committed affordance markers, exempt from time-based decay per Section 4.5, are:

- `AffordanceMarker(warehouse-south, warehouse_matching)` – intensity 1.0.
- `AffordanceMarker(cost-estimator, deal_estimation)` – intensity 1.0.

Non-competitive agents (Call Transcription, Deal Summary, Client Necessity, Compliance, Proposal Template) do not produce affordance markers because they are invoked directly without bidding (Section 4.5). These markers bias bid scoring in subsequent runs through the trust feedback channel of Eq. 5, reinforcing the successful assignments.

A.7 Audit and Traceability

All phases above write structured records into the common audit stream described in Section 4.10, correlated by the Pipeline and Run IDs given in the Task Normalization subsection. Because this run completes validated planning, mandatory human approval, ethics review, and execution without triggering any Tier 3 escalation, it receives the

maximum traceability score for framework configuration C0.

B. Failure Recovery Trace: C0

This appendix provides a step-by-step trace of a single injected failure event under configuration C0, illustrating how the three-tier recovery mechanism defined in Section 4.9 (Algorithm 6) activates in practice. The trace corresponds to one of the ten C0 injections aggregated in Table 30.

B.1 Failure Injection

- **Configuration**: C0 (full framework)
- **Client**: FreshCo
- **Failure injected at**: warehouse-match execution stage
- **Pipeline ID**: `f15dade6-b0d1-4f78-8732-d258c8966a18`
- **Timeout budget**: 300 s

B.2 Recovery Sequence

The three-tier recovery mechanism activates in the order defined by Algorithm 6.

1. **Detection.** The Task Execution Orchestrator detects the `task-failed` event over RabbitMQ and hands control to the Retry Middleware.

2. **Tier 1 — Bounded Local Retries.** The Retry Middleware attempts re-execution under the configured retry policy with exponential backoff, up to the default maximum of three attempts. This tier is intended to absorb transient faults such as network glitches and intermittent timeouts without escalation. In this run, all three attempts fail and the failure is propagated to Tier 2.

3. **Tier 2 — Adaptive Replanning (with concurrent pheromone penalty).** On retry exhaustion, the LLM Strategist (Section 4.8) inspects the current task graph and execution state and proposes a graph delta that re-routes the failed task to an alternative agent with compatible capabilities. The proposed delta is validated against the framework's structural invariants, namely acyclicity, dependency preservation, and exactly one assignment per task, before being committed to the coordination graph store. Concurrently with the Strategist action, the Pheromone Engine applies a penalty update (Eq. 6) to the failed agent's `AffordanceMarker` intensity, biasing all future assignments away from that agent.

4. **Tier 3 — Structured Escalation (not reached).** Had Tier 2 failed to resolve the event, the system would have raised a structured escalation record containing the run and plan identifiers, a machine-readable failure reason, the severity level derived from the plan's risk classification, recent execution summaries, and an indication that automated recovery had been exhausted, and would have routed it to a human operator through the monitoring and alerting channels. Tier 3 was **not activated** in this run because Tier 2 adaptive replanning resolved the failure on the first proposed delta.

B.3 Timing Breakdown

The total recovery time of 62 s decomposes into instrumented components and uninstrumented residual time. The two components measured directly are reported below. The remainder is attributable to Tier 1 retry attempts with exponential backoff, the concurrent pheromone penalty write, message-broker propagation, and re-execution of the task on the alternative agent.

- **LLM Strategist call**: 18.0 s total
  - Prompt assembly: 1.2 s
  - GPT-4o-mini structured-output inference: 16.4 s
  - Response parsing: 0.4 s
- **Graph-delta validation**: 1.8 s (acyclicity, dependency preservation, and single-assignment checks all passed on the first proposal)
- **Total recovery time**: 62 s

B.4 Outcome

- **Recovery successful**: Yes
- **Resolving tier**: Tier 2 (adaptive replanning)
- **Final pipeline status**: `COMPLETED`, 12/12 tasks completed
- **Persistent effect**: pheromone penalty recorded against the failed agent's `AffordanceMarker`, reducing its priority in subsequent bidding rounds

C. Wilcoxon Signed-Rank Statistics

This appendix reports the raw Wilcoxon signed-rank statistics referenced in Chapter 6. Each test pairs the per-client mean accuracy of C0 against that of an ablation configuration C_x over the five evaluation clients, yielding $N = 5$ paired observations per test. The Bonferroni-corrected threshold is $\alpha_{\text{Bonf}} = 0.05/8 = 0.0063$. All comparisons are non-significant, as expected under a five-client paired design with accuracy near the ceiling; the main-text discussion proceeds to Cliff's delta (Table 22) as a power-independent alternative.

Comparison	W	p (raw)	p (Bonferroni)	Significant
C0 vs. C1	0.00	1.000	n.s.	No
C0 vs. C2	0.00	0.125	n.s.	No
C0 vs. C3	0.00	0.500	n.s.	No
C0 vs. C3+C4	0.00	1.000	n.s.	No
C0 vs. C4	0.00	0.500	n.s.	No
C0 vs. C5	0.00	1.000	n.s.	No
C0 vs. C6	0.00	1.000	n.s.	No
C0 vs. C7	0.00	0.250	n.s.	No

Table 39: Wilcoxon signed-rank tests comparing C0 against each ablation configuration. Pairs are per-client mean accuracies over the five evaluation clients ($N = 5$); Bonferroni-corrected at $\alpha_{\text{Bonf}} = 0.0063$.

D. User Perception Survey

This appendix contains the full questionnaire used in the user evaluation study described in section 6.3. The survey was administered via Google Forms. All responses were collected anonymously in compliance with GDPR guidelines [3].

GDPR-Compliant Consent

Before beginning the survey, participants were required to confirm each of the following statements:

a) I have read and understood the data protection information above.

b) I understand that my participation is voluntary and I can withdraw at any time.

c) I consent to my anonymised responses being used for academic research.

d) I understand that no personally identifiable information will be published.

Participant Profile

1. **Which group best describes your profile?**
   Single choice.
   - Domain Expert (logistics, warehouse brokerage, supply chain professional)
   - Technical Evaluator (software engineer, AI/ML practitioner, researcher)
2. **Years of Professional Experience**
   Single choice.
   - Less than 2 years
   - 2–5 years
   - 5–10 years
   - More than 10 years
3. **How familiar are you with AI or large language model tools (e.g., ChatGPT, copilots)?**
   Single choice.
   - Not familiar at all
   - Somewhat familiar (I've used them casually)
   - Familiar (I use them regularly in my work)
   - Very familiar (I build or research AI systems)

Section 1: Output Quality

All items in this section use a 5-point Likert scale: 1 = Strongly Disagree, 2 = Disagree, 3 = Neutral, 4 = Agree, 5 = Strongly Agree.

Scenario 1 — FreshCo (Cold-Storage, Milan)

4. The warehouse suggestion matches the client's stated requirements (location, budget, timeline, storage type).
5. The compliance check identified the correct regulatory and operational constraints.
6. The generated proposal is professional enough to send to a client as a first draft.
7. The recommendation includes information I wouldn't have considered on my own.

Scenario 2 — GreenLeaf (Hazmat / Critical Case)

8. The warehouse suggestion matches the client's stated requirements.
9. The compliance check identified the correct regulatory and operational constraints.
10. The generated proposal is professional enough to send to a client as a first draft.
11. The recommendation includes information I wouldn't have considered on my own.

Scenario 3 — Failure Recovery Case

12. The system handled the failure scenario in a reasonable way.
13. The fallback recommendation was still useful despite the primary failure.
14. **Would you trust these outputs as a first draft for a real client engagement? Why or why not?**
    Free text.
15. **What is missing from these recommendations that you would add manually?**
    Free text.

Section 2: Decision Transparency

5-point Likert scale (Strongly Disagree – Strongly Agree).

16. I can understand why a particular warehouse was selected over alternatives.
17. The bidding scores make the agent selection process transparent.
18. The audit trail provides enough information to explain the decision to a client or stakeholder.
19. If the output contained an error, I could identify where in the pipeline it occurred.
20. **What information is missing from the decision trace that would improve your understanding?**
    Free text.

Section 3: Framework Value

This section was shown only to Technical Evaluators. 5-point Likert scale (Strongly Disagree – Strongly Agree).

21. The Multi-Agent approach adds value over a single monolithic LLM call.
22. The automatic failure recovery (fallback to next-best agent) is a useful mechanism.
23. The SHACL constraints provide meaningful safety guarantees for the outputs.
24. The pheromone-based adaptation is a practical mechanism for dynamic agent selection.
25. **In what scenarios would you choose this framework over a simpler orchestration tool (e.g. LangChain, a single prompt chain)?**
    Free text.

Section 4: Comparative Evaluation

Participants were shown three anonymised outputs (Output A, Output B, Output C) produced by different system configurations.

26. **Which output would you trust most for a real client decision?**
    Single choice: Output A / Output B / Output C.
27. **Which output is easiest to audit if something goes wrong?**
    Single choice: Output A / Output B / Output C.
28. **Rank the three outputs by overall quality** (1 = best, 3 = worst).
    Ranking for each: Output A, Output B, Output C.
29. **What made you prefer your top-ranked output?**
    Free text.

Final Thoughts

30. **Overall, how would you rate the usefulness of this framework for warehouse brokerage?**
    5-point scale: 1 = Not useful at all, . . . , 5 = Extremely useful.
31. **Any additional comments, suggestions, or concerns?**
    Free text.

E. Survey Materials: Comparative Evaluation

Client Request (identical for all three approaches)

- **FreshCo Logistics** — Food & Beverage company (Maria Rossi, `maria@freshco.eu`) requires a cold-storage warehouse in the Milan metropolitan area.

- **Space** — 15,000–25,000 sqft
- **Budget** — €18,000/month (firm)
- **Lease** — 36 months
- **Move in** — June 1, 2026
- **Requirements** — cold storage, food-grade flooring, temperature monitoring
- **Priority** — High

E.1 Output A

Based on the requirements, I recommend **LogisPark Milano Nord** for FreshCo Logistics.

Why this warehouse:

- Located in Milan with 22,000 sqft available.
- Has cold storage capabilities.
- Monthly rent would be approximately €16,500 (22,000 × €0.75/sqft), within the €18,000 budget.
- Modern facility built in 2019 with enhanced security.
- 8 dock doors and 28 ft ceiling height.

Draft proposal:

> Dear Maria,
>
> We would like to recommend LogisPark Milano Nord for your cold storage needs. The facility offers 22,000 square feet with cold storage capabilities at €16,500 per month, well within your budget of €18,000. The warehouse was built in 2019 and features enhanced security and 8 dock doors.
>
> The facility can support your June 2026 move-in timeline. We suggest scheduling a site visit to confirm the cold storage temperature range meets your −25 °C requirement and to assess the dock configuration for your refrigerated truck fleet.
>
> Please let us know if you would like to proceed with a site visit.
>
> Best regards

E.2 Output B

Warehouse Recommendation: FreshCo Logistics

Assigned agent: Warehouse Central Europe.

Analysis of Central European warehouses:

Warehouse	City	Sqft	Cold?	Monthly Cost	Match
Bavaria Logistics Hub	Munich	40,000	No	€34,000	NO — no cold storage, wrong city
Stuttgart TechCenter	Stuttgart	35,000	No	€31,500	NO — no cold storage, wrong city
PharmaStore Lyon	Lyon	28,000	Yes	€26,600	PARTIAL — wrong country, over budget
Paris CDG Logistics	Roissy	25,000	Yes	€27,500	PARTIAL — wrong country, over budget

No warehouses in the Central Europe region match the client's requirements. The client needs a Milan-area facility, which is outside this agent's coverage area. PharmaStore Lyon and Paris CDG have cold storage but are in France (wrong location) and exceed the €18,000 budget.

Match Score: 0 / 100 for location-eligible options.

Recommendation: Unable to provide a suitable recommendation. This request should be routed to the Southern Europe specialist.

Timeline Analysis: FreshCo Logistics

Assigned agent: Speed Estimator.

Milestone	Estimated Date
Lease signing	April 14, 2026
Food-grade flooring install	May 15, 2026
Cold storage verification	May 22, 2026
Operational	June 1, 2026

Assessment: timeline is feasible with a 6-week lead time. No critical path risks identified.

Compliance Report

```
RESULT: NO VALID PAIRING TO CHECK
The warehouse agent did not produce a viable recommendation.
Compliance check skipped -- no warehouse-client pairing to validate.
```

Proposal

Unable to generate a complete proposal. The assigned warehouse agent could not find a match in its region. Manual intervention required to re-route to the correct regional agent.

E.3 Output C

Agent Selection Summary

E.3.0.1 Warehouse matching - competitive bidding.

Agent	capMatch	Trust	Cost-effectiveness	Regional-tag bonus	Bid Score
Warehouse South	1.00	0.82	0.78	+0.28 (Milan region)	**0.699**
Warehouse Central	1.00	0.78	0.62	0.00	0.339
Warehouse North	1.00	0.75	0.70	0.00	0.320

Winner: Warehouse South (0.699).

E.3.0.2 Cost vs. speed estimator - competitive bidding.

Agent	capMatch	Trust	Cost-effectiveness	Budget-match bonus	Bid Score
Cost Estimator	1.00	0.80	0.80	+0.10 (firm budget)	**0.696**
Speed Estimator	1.00	0.78	0.65	0.00	0.320

Winner: Cost Estimator (0.696). FreshCo has high priority but a firm €18,000/month budget, which aligns with the cost-optimised strategy.

Warehouse Recommendation: LogisPark Milano Nord

Criterion	Requirement	LogisPark Milano Nord	Match
Location	Milan metropolitan area	Via Industriale 42, Milan	✓
Available space	15,000–25,000 sqft	22,000 sqft	✓
Ceiling height	≥ 24 ft	28 ft	✓
Dock doors	≥ 6	8	✓
Cold storage	Required	Yes ($-30\,^{\circ}$C capable)	✓
Monthly cost	$\leq$ €18,000	€16,500	✓
Security	—	Enhanced	Bonus

Match Score: 95 / 100.

Strengths:

- Cold zone supports $-30\,^{\circ}$C (exceeds $-25\,^{\circ}$C requirement).
- 8,000 sqft dedicated cold area sufficient for frozen goods.
- Landlord offers €50,000 fit-out allowance for food-grade flooring.

Gaps:

- 8 dock doors vs. 12 refrigerated trucks requested - confirm dock scheduling at site visit.
- Temperature monitoring system not confirmed in database - verify during site visit.

Compliance Report

Rule	Result	Detail
COLD_STORAGE_REQUIRED	PASS	LogisPark has cold storage
HAZMAT_REQUIRED	N/A	Not required by client
HIGH_SECURITY_FOR_CRITICAL	N/A	Priority is high, not critical
SQFT_WITHIN_RANGE	PASS	22,000 sqft within 15,000–25,000
BUDGET_COMPLIANCE	PASS	€16,500/mo under €18,000 budget

Result: 5/5 checks passed, no violations, no warnings.

Cost Analysis

Cost Component	Estimate	Notes
Monthly rent	€16,500	22,000 sqft × €0.75/sqft
Total lease (36 months)	€594,000	
Food-grade flooring	€45,000	Offset by landlord allowance
Temperature monitoring system	€12,000	One-time install
Landlord fit-out allowance	−€50,000	Negotiated
Total cost of occupancy	**€601,000**	Over 36-month lease
Effective monthly cost	**€16,694**	€1,306/mo headroom vs. budget

Client Proposal

Dear Maria,

Following our analysis of your requirements for a cold-storage facility in the Milan area, we are pleased to recommend **LogisPark Milano Nord** as your primary option.

Property summary:

- Location: Via Industriale 42, Milan (Lombardy).
- Available space: 22,000 sqft with 28 ft ceiling clearance.
- Cold storage: −30 °C capable zone (8,000 sqft), exceeding your −25 °C requirement.
- Dock access: 8 dock doors with enhanced security.
- Monthly rent: €16,500 (within your €18,000 budget).

Financial terms:

- 36-month lease: €594,000 total.
- Landlord-funded fit-out allowance: €50,000 (covers food-grade flooring upgrade).
- Estimated out-of-pocket setup: €7,000 (temperature monitoring system).
- Effective monthly cost: €16,694.

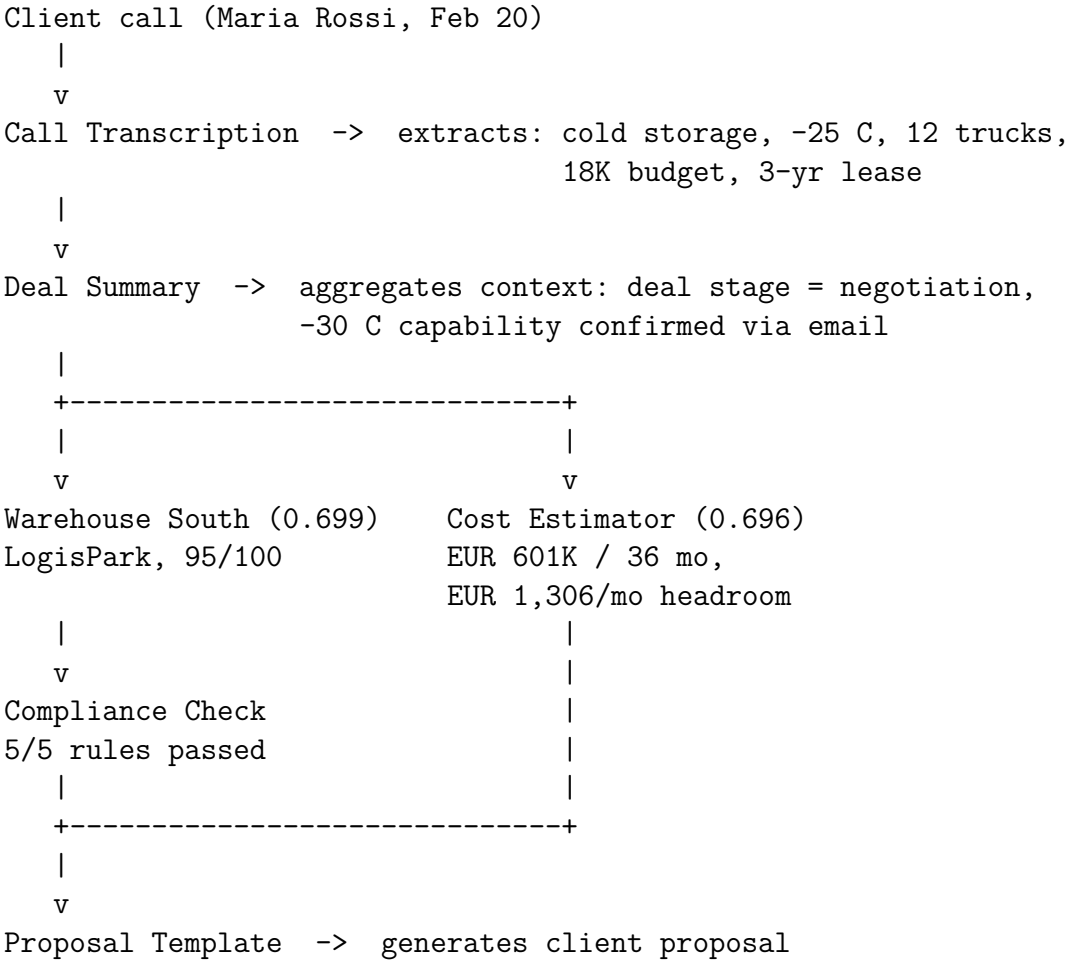
Recommended next steps:

1. Site visit (available week of March 31).
2. Confirm dock door scheduling for 12 refrigerated trucks.
3. Temperature monitoring system specification review.

We are confident this property meets your operational needs while remaining within budget. Please let us know your availability for a site visit.

Best regards,
[Brokerage Name]

Decision Trace



All bids, scores, compliance checks, and cost calculations are stored in the audit log for post-hoc inspection. Any step can be replayed or explained on demand.

Bibliography

[1] Reid G. Smith. "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver". In: *IEEE Transactions on Computers* 29.12 (1980), pp. 1104–1113. DOI: 10.1109/TC.1980.1675516.

[2] European Union. *Regulation (EU) 2024/1689 (Artificial Intelligence Act)*. Official Journal of the European Union. [Online]. Accessed: Feb. 14, 2026. 2024. URL: https://eur-lex.europa.eu/eli/reg/2024/1689/oj/eng.

[3] European Union. *Regulation (EU) 2016/679 (General Data Protection Regulation)*. Official Journal of the European Union. [Online]. Accessed: Apr. 17, 2026. 2016. URL: https://eur-lex.europa.eu/eli/reg/2016/679/oj.

[4] OpenAI. *GPT-4o mini: Advancing Cost-Efficient Intelligence*. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/. [Online]. Accessed: Apr. 23, 2026. 2024.

[5] Jiarui Zhang, Gang Wang, and Yafei Song. "Task Assignment of the Improved Contract Net Protocol under a Multi-Agent System". In: *Algorithms* 12.4 (2019), p. 70. DOI: 10.3390/a12040070.

[6] Martin J. Dürst and Michel Suignard. *Internationalized Resource Identifiers (IRIs)*. RFC 3987. IETF, 2005. DOI: 10.17487/RFC3987.

[7] Couchbase, Inc. *N1QL: Query Language for JSON Documents*. https://docs.couchbase.com/server/current/n1ql/n1ql-language-reference/index.html. [Online]. Accessed: Apr. 23, 2026. 2024.

[8] National Institute of Standards and Technology. *Artificial Intelligence Risk Management Framework (AI RMF) 1.0*. Tech. rep. NIST, 2023. DOI: 10.6028/NIST.AI.100-1.

[9] W3C. *OWL 2 Web Ontology Language Document Overview (Second Edition)*. W3C Recommendation. [Online]. Accessed: Feb. 14, 2026. 2012. URL: https://www.w3.org/TR/owl2-overview/.

[10] W3C. *PROV-O: The PROV Ontology*. W3C Recommendation. [Online]. Accessed: Feb. 14, 2026. 2013. URL: https://www.w3.org/TR/prov-o/.

[11] W3C. *RDF 1.1 Concepts and Abstract Syntax*. W3C Recommendation. [Online]. Accessed: Feb. 14, 2026. 2014. URL: https://www.w3.org/TR/rdf11-concepts/.

[12] United Nations. *The 17 Goals — Sustainable Development*. Online. [Online]. Accessed: Mar. 23, 2026. 2015. URL: https://sdgs.un.org/goals.

[13] W3C. *Shapes Constraint Language (SHACL)*. W3C Recommendation. [Online]. Accessed: Feb. 14, 2026. 2017. URL: https://www.w3.org/TR/shacl/.

[14] W3C. *SPARQL 1.1 Query Language*. W3C Recommendation. [Online]. Accessed: Feb. 14, 2026. 2013. URL: https://www.w3.org/TR/sparql11-query/.

[15] Francis Heylighen. "Stigmergy as a Universal Coordination Mechanism I: Definition and Components". In: *Cognitive Systems Research* 38 (2016), pp. 4–13. DOI: 10.1016/j.cogsys.2015.12.002.

[16] Michael Wooldridge and Nicholas R. Jennings. "Intelligent Agents: Theory and Practice". In: *The Knowledge Engineering Review* 10.2 (1995), pp. 115–152. DOI: 10.1017/S0269888900008122.

[17] Xinyi Li et al. “A Survey on LLM-Based Multi-Agent Systems: Workflow, Infrastructure, and Challenges”. In: *Vicinagearth* 1 (2024), p. 9. DOI: 10.1007/s44336-024-00009-2.

[18] LangChain AI. *LangGraph*. GitHub repository. [Online]. Accessed: Feb. 14, 2026. 2024. URL: https://github.com/langchain-ai/langgraph.

[19] CrewAIInc. *CrewAI*. GitHub repository. [Online]. Accessed: Feb. 14, 2026. 2024. URL: https://github.com/crewAIInc/crewAI.

[20] Qingyun Wu et al. “AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation”. In: *arXiv preprint arXiv:2308.08155* (2023). [Online]. Accessed: Feb. 14, 2026.

[21] Torsten Spieldenner and Melvin Chelli. “Linked Data as Stigmergic Medium for Decentralized Coordination”. In: *Proc. 16th Int. Conf. on Software Technologies (ICSOFT)*. SCITEPRESS, 2021, pp. 347–357. DOI: 10.5220/0010518003470357.

[22] Tim Berners-Lee, James Hendler, and Ora Lassila. “The Semantic Web”. In: *Scientific American* 284.5 (2001), pp. 34–43. URL: https://www.scientificamerican.com/article/the-semantic-web/. [Online]. Accessed: Feb. 14, 2026.

[23] OWASP. *OWASP Top 10 for Large Language Model Applications*. [Online]. Accessed: Feb. 14, 2026. 2024. URL: https://owasp.org/www-project-top-10-for-large-language-model-applications/.

[24] Michael Wooldridge. *An Introduction to MultiAgent Systems*. 2nd ed. Chichester, UK: John Wiley & Sons, 2009.

[25] Patrick Th. Eugster et al. “The Many Faces of Publish/Subscribe”. In: *ACM Computing Surveys* 35.2 (2003), pp. 114–131.

[26] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 4th ed. Hoboken, NJ: Pearson, 2020. ISBN: 978-0134610993.

[27] Barbara Hayes-Roth. “A Blackboard Architecture for Control”. In: *Artificial Intelligence* 26.3 (1985), pp. 251–321.

[28] Alan H. Bond and Les Gasser, eds. *Readings in Distributed Artificial Intelligence*. San Mateo, CA: Morgan Kaufmann, 1988.

[29] Anand S. Rao and Michael P. Georgeff. “BDI Agents: From Theory to Practice”. In: *Proc. 1st Int. Conf. on Multi-Agent Systems (ICMAS)*. 1995.

[30] Tim Finin et al. “KQML as an Agent Communication Language”. In: *Proc. 3rd Int. Conf. on Information and Knowledge Management (CIKM)*. 1994, pp. 456–463.

[31] Foundation for Intelligent Physical Agents. *FIPA ACL Message Structure Specification*. Tech. rep. SC00061G. [Online]. Accessed: Feb. 14, 2026. FIPA, 2002. URL: http://www.fipa.org/specs/fipa00061/.

[32] Fabio Luigi Bellifemine, Giovanni Caire, and Dominic Greenwood. *Developing Multi-Agent Systems with JADE*. Chichester, UK: John Wiley & Sons, 2007. ISBN: 978-0470057476.

[33] Lucian Buşoniu, Robert Babuška, and Bart De Schutter. “A Comprehensive Survey of Multiagent Reinforcement Learning”. In: *IEEE Trans. Syst., Man, Cybern., Part C* 38.2 (2008), pp. 156–172.

[34] Nicola Mc Donnell. “Leveraging Nature-inspired Techniques to Semi-automate the Design of Multi-Agent Systems”. [Online]. Accessed: Feb. 14, 2026. Ph.D. dissertation. University of Galway, 2023. URL: https://hdl.handle.net/10379/17838.

[35] Shunyu Yao et al. *ReAct: Synergizing Reasoning and Acting in Language Models*. [Online]. Accessed: Feb. 14, 2026. 2022. arXiv: 2210.03629 [cs.CL]. URL: https://arxiv.org/abs/2210.03629.

[36] Nazmus Ashrafi, Salah Bouktif, and Mohammed Mediani. “Enhancing LLM Code Generation: A Systematic Evaluation of Multi-Agent Collaboration and Runtime Debugging for Improved Accuracy, Reliability, and Latency”. In: *arXiv preprint arXiv:2505.02133* (2025). [Online]. Accessed: Feb. 14, 2026. DOI: 10.48550/arXiv.2505.02133.

[37] Guohao Li et al. “CAMEL: Communicative Agents for “Mind” Exploration of Large Language Model Society”. In: *arXiv preprint arXiv:2303.17760* (2023). [Online]. Accessed: Feb. 14, 2026. DOI: 10.48550/arXiv.2303.17760.

[38] Noah Shinn et al. *Reflexion: Language Agents with Verbal Reinforcement Learning*. [Online]. Accessed: Feb. 14, 2026. 2023. arXiv: 2303.11366 [cs.AI]. URL: https://arxiv.org/abs/2303.11366.

[39] Microsoft. *JARVIS: A System to Connect Large Language Models with the ML Community*. GitHub repository. [Online]. Accessed: Feb. 14, 2026. URL: https://github.com/microsoft/JARVIS.

[40] Yongliang Shen et al. “HuggingGPT: Solving AI Tasks with ChatGPT and Its Friends in HuggingFace”. In: *Advances in Neural Information Processing Systems*. [Online]. Accessed: Feb. 14, 2026. 2023.

[41] Significant-Gravitas. *AutoGPT: A Python/TypeScript Open-Source Autonomous AI Agent Platform*. GitHub repository. [Online]. Accessed: Feb. 14, 2026. 2024. URL: https://github.com/Significant-Gravitas/AutoGPT.

[42] Joon Sung Park et al. “Generative Agents: Interactive Simulacra of Human Behavior”. In: *arXiv preprint arXiv:2304.03442* (2023). [Online]. Accessed: Feb. 14, 2026. DOI: 10.48550/arXiv.2304.03442.

[43] Lakshmi Varanasi. *PwC Launches a New Platform for AI Agents: Agent OS*. Business Insider. [Online]. Accessed: Feb. 14, 2026. 2025. URL: https://www.businessinsider.com/pwcs-launches-a-new-platform-for-ai-agents-agent-os-2025-3.

[44] Saleema Amershi et al. “Guidelines for Human-AI Interaction”. In: *Proc. 2019 CHI Conf. on Human Factors in Computing Systems (CHI ’19)*. 2019.

[45] Franz Baader et al., eds. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge, UK: Cambridge University Press, 2003.

[46] Tim Berners-Lee. *Linked Data*. W3C Design Issues Note. [Online]. Accessed: Feb. 14, 2026. 2006. URL: https://www.w3.org/DesignIssues/LinkedData.html.

[47] Alistair Miles and Sean Bechhofer. *SKOS Simple Knowledge Organization System Reference*. W3C Recommendation. [Online]. Accessed: Feb. 14, 2026. Aug. 2009. URL: https://www.w3.org/TR/2009/REC-skos-reference-20090818/.

[48] H. Van Dyke Parunak. “A Survey of Environments and Mechanisms for Human-Human Stigmergy”. In: *Environments for Multi-Agent Systems II*. Vol. 3830. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2006, pp. 163–186.

[49] C. M. Waters and B. L. Bassler. “Quorum Sensing: Cell-to-Cell Communication in Bacteria”. In: *Annual Review of Cell and Developmental Biology* 21 (2005), pp. 319–346.

[50] Amira Rezgui and Kevin Crowston. “Stigmergic Coordination in Wikipedia”. In: *Proc. 14th Int. Symp. on Open Collaboration (OpenSym '18)*. 2018. DOI: 10.1145/3233391.3233543.

[51] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. New York: Oxford University Press, 1999. ISBN: 978-0195131598.

[52] Martin Fowler and James Lewis. *Microservices*. [Online]. Accessed: Feb. 14, 2026. 2014. URL: https://martinfowler.com/articles/microservices.html.

[53] Chris Richardson. *Pattern: Messaging*. Online. [Online]. Accessed: Feb. 14, 2026. 2018. URL: https://microservices.io/patterns/communication-style/messaging.html.

[54] Gregor Hohpe and Bobby Woolf. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Boston, MA: Addison-Wesley, 2003. ISBN: 978-0321200686.

[55] Apache Airflow. *DAGs — Airflow Documentation*. Online documentation. [Online]. Accessed: Feb. 14, 2026. 2024. URL: https://airflow.apache.org/docs/apache-airflow/stable/core-concepts/dags.html.

[56] Amazon Web Services. *What Is AWS Step Functions?* Online documentation. [Online]. Accessed: Feb. 14, 2026. 2024. URL: https://docs.aws.amazon.com/step-functions/latest/dg/welcome.html.

[57] Apache Airflow. *Scheduling and Triggers — Airflow Documentation*. Online documentation. [Online]. Accessed: Feb. 14, 2026. 2024. URL: https://airflow.apache.org/docs/apache-airflow/stable/authoring-and-scheduling/index.html.

[58] Confluent. *Client Configuration Settings for Confluent Cloud*. Online documentation. [Online]. Accessed: Feb. 14, 2026. 2024. URL: https://docs.confluent.io/cloud/current/client-apps/client-configs.html.

[59] Amazon Web Services. *Idempotency — AWS Lambda Powertools for Python Documentation*. Online documentation. [Online]. Accessed: Feb. 14, 2026. 2024. URL: https://docs.aws.amazon.com/lambda-powertools-python/latest/utilities/idempotency/.

[60] Stripe. *Idempotent Requests — Stripe API Reference*. Online documentation. [Online]. Accessed: Feb. 14, 2026. 2024. URL: https://docs.stripe.com/api/idempotent_requests.

[61] Jim Gray and Andreas Reuter. *Transaction Processing: Concepts and Techniques*. San Mateo, CA: Morgan Kaufmann, 1993.

[62] Theo Härder and Andreas Reuter. “Principles of Transaction-Oriented Database Recovery”. In: *ACM Computing Surveys* 15.4 (1983), pp. 287–317.

[63] Hector Garcia-Molina and Kenneth Salem. *SAGAS*. Tech. rep. CS-TR-070-87. Princeton University, Department of Computer Science, 1987.

[64] Chris Richardson. *Pattern: Transactional Outbox*. Online. [Online]. Accessed: Feb. 14, 2026. 2018. URL: https://microservices.io/patterns/data/transactional-outbox.html.

[65] Brian P. Gerkey and Maja J. Matarić. “A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems”. In: *The International Journal of Robotics Research* 23.9 (2004), pp. 939–954. DOI: 10.1177/0278364904045564.

[66] M. B. Dias et al. “Market-Based Multirobot Coordination: A Survey and Analysis”. In: *Proceedings of the IEEE* 94.7 (2006), pp. 1257–1270. DOI: 10.1109/JPROC.2006.876939.

[67] George Marios Skaltsis, Hyo-Sang Shin, and Antonios Tsourdos. “A Survey of Task Allocation Techniques in MAS”. In: *2021 Int. Conf. on Unmanned Aircraft Systems (ICUAS)*. Athens, Greece, 2021, pp. 488–497.

[68] Abraham Silberschatz, Peter B. Galvin, and Greg Gagne. *Operating System Concepts*. 10th ed. Hoboken, NJ: John Wiley & Sons, 2018.

[69] Yohei Nakajima. *BabyAGI*. GitHub repository. [Online]. Accessed: Feb. 14, 2026. 2023. URL: https://github.com/yoheinakajima/babyagi.

[70] Chen Qian et al. “ChatDev: Communicative Agents for Software Development”. In: *arXiv preprint arXiv:2307.07924* (2023). [Online]. Accessed: Feb. 14, 2026.

[71] Sirui Hong et al. “MetaGPT: Meta Programming for a Multi-Agent Collaborative Framework”. In: *arXiv preprint arXiv:2308.00352* (2023). [Online]. Accessed: Feb. 14, 2026.

[72] LangChain. *LangGraph Overview*. [Online]. Accessed: Feb. 14, 2026. 2025. URL: https://docs.langchain.com/oss/python/langgraph/overview.

[73] LangChain. *LangGraph Runtime (Pregel)*. [Online]. Accessed: Feb. 14, 2026. 2025. URL: https://docs.langchain.com/oss/python/langgraph/pregel.

[74] Grzegorz Malewicz et al. “Pregel: A System for Large-Scale Graph Processing”. In: *Proc. 2010 ACM SIGMOD Int. Conf. on Management of Data*. ACM, 2010, pp. 135–146.

[75] LangChain. *LangGraph Persistence (Checkpointers)*. [Online]. Accessed: Feb. 14, 2026. 2025. URL: https://docs.langchain.com/oss/python/langgraph/persistence.

[76] CrewAI. *Flows — CrewAI Documentation*. [Online]. Accessed: Feb. 14, 2026. 2025. URL: https://docs.crewai.com/en/concepts/flows.

[77] Pierre-Paul Grassé. “La reconstruction du nid et les coordinations interindividuelles chez Bellicositermes natalensis et Cubitermes sp.: la théorie de la stigmergie”. In: *Insectes Sociaux* 6 (1959), pp. 41–80. DOI: 10.1007/BF02223791.

[78] Brendan Burns. *Designing Distributed Systems: Patterns and Paradigms for Scalable, Reliable Services*. Sebastopol, CA: O’Reilly Media, 2018. ISBN: 978-1491983645.

[79] Tim Berners-Lee, Roy T. Fielding, and Larry Masinter. *Uniform Resource Identifier (URI): Generic Syntax*. RFC 3986. IETF, 2005. DOI: 10.17487/RFC3986.

[80] Marco Dorigo and Thomas Stützle. *Ant Colony Optimization*. Cambridge, MA: MIT Press, 2004. ISBN: 978-0262042192.

[81] Arthur B. Kahn. “Topological Sorting of Large Networks”. In: *Communications of the ACM* 5.11 (1962), pp. 558–562. DOI: 10.1145/368996.369025.

[82] Thomas H. Cormen et al. *Introduction to Algorithms*. 3rd ed. Cambridge, MA: MIT Press, 2009.

[83] W3C. *Trace Context*. W3C Recommendation. [Online]. Accessed: Feb. 14, 2026. 2021. URL: https://www.w3.org/TR/trace-context/.

[84] OECD. *OECD AI Principles*. [Online]. Accessed: Feb. 14, 2026. 2019. URL: https://oecd.org/en/topics/ai-principles.html.

[85] Couchbase, Inc. *Couchbase Server Documentation*. https://docs.couchbase.com/server/current/introduction/intro.html. [Online]. Accessed: Apr. 23, 2026. 2024.

[86] Apache Software Foundation. *Apache Jena Fuseki: A SPARQL Server*. Online documentation. [Online]. Accessed: Apr. 19, 2026. 2024. URL: https://jena.apache.org/documentation/fuseki2/.

[87] HashiCorp. *Consul: Service Discovery and Service Mesh*. https://developer.hashicorp.com/consul/docs. [Online]. Accessed: Apr. 23, 2026. 2024.

[88] OpenTelemetry Authors. *OpenTelemetry: Observability Framework for Cloud-Native Software*. https://opentelemetry.io/docs/. [Online]. Accessed: Apr. 23, 2026. 2024.

[89] Sebastián Ramírez. *FastAPI: Modern, Fast Web Framework for Building APIs with Python*. https://fastapi.tiangolo.com. [Online]. Accessed: Apr. 23, 2026. 2024.

[90] Pydantic Authors. *Pydantic: Data Validation Using Python Type Hints*. https://docs.pydantic.dev. [Online]. Accessed: Apr. 23, 2026. 2024.

[91] ISO/IEC. *ISO/IEC 42001:2023 — Information Technology — Artificial Intelligence — Management System*. [Online]. Accessed: Feb. 14, 2026. 2023. URL: https://www.iso.org/standard/42001.

[92] Frank Wilcoxon. "Individual Comparisons by Ranking Methods". In: *Biometrics Bulletin* 1.6 (1945), pp. 80–83. DOI: 10.2307/3001968.

[93] Carlo E. Bonferroni. "Teoria statistica delle classi e calcolo delle probabilità". In: *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commerciali di Firenze* 8 (1936), pp. 3–62.

[94] Jeanine Romano et al. "Appropriate Statistics for Ordinal Level Data: Should We Really Be Using t-Test and Cohen's d for Evaluating Group Differences on the NSSE and Other Surveys?" In: *Annual Meeting of the Florida Association of Institutional Research* (2006), pp. 1–33.

[95] Karl Pearson. "On the Criterion that a Given System of Deviations from the Probable in the Case of a Correlated System of Variables is Such that it Can be Reasonably Supposed to have Arisen from Random Sampling". In: *Philosophical Magazine Series 5* 50.302 (1900), pp. 157–175. DOI: 10.1080/14786440009463897.

[96] Harald Cramér. *Mathematical Methods of Statistics*. Princeton, NJ: Princeton University Press, 1946.

[97] Milton Friedman. "The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance". In: *Journal of the American Statistical Association* 32.200 (1937), pp. 675–701. DOI: 10.1080/01621459.1937.10503522.

[98] Henry B. Mann and Donald R. Whitney. "On a Test of Whether One of Two Random Variables is Stochastically Larger than the Other". In: *The Annals of Mathematical Statistics* 18.1 (1947), pp. 50–60. DOI: 10.1214/aoms/1177730491.

[99] Robert Rosenthal. *Meta-Analytic Procedures for Social Research*. Revised. Newbury Park, CA: SAGE Publications, 1991. DOI: 10.4135/9781412984997.

[100] Universitat Politècnica de Catalunya. *Guia per incorporar l'anàlisi de sostenibilitat i implicacions ètiques en els TFE*. Informative document presented to the Academic Policy Commission, Jun. 28, 2023. [Online]. Accessed: Mar. 23, 2026. 2023. URL: https://govern.upc.edu.

[101] Electricity Maps. *Electricity Grid Review 2025: Spain*. Online report. Spanish grid (ES) data, 2025. [Online]. Accessed: Feb. 14, 2026. 2025. URL: https://www.electricitymaps.com/grid-in-review-2025/spain.

[102] Kadan Lottick et al. "Energy Usage Reports: Environmental Awareness as Part of Algorithmic Accountability". In: *NeurIPS Workshop on Tackling Climate Change with Machine Learning*. [Online]. Accessed: Feb. 14, 2026. Vancouver, Canada, 2019. URL: https://arxiv.org/abs/1911.08354.

[103] Couchbase. *Document Expiration (Time-To-Live)*. Online documentation. [Online]. Accessed: Feb. 14, 2026. 2025. URL: https://docs.couchbase.com/.

[104] Kubernetes Documentation. *Horizontal Pod Autoscaling*. Online documentation. [Online]. Accessed: Feb. 14, 2026. 2025. URL: https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/.

[105] Universitat Politècnica de Catalunya. *Code of Ethics of the Universitat Politècnica de Catalunya*. Governing Council Decision CG/2022/02. [Online]. Accessed: Apr. 23, 2026. Feb. 2022. URL: https://talenthub.upc.edu/en/docs/upc-ethics.pdf.

[106] Sandra Wachter, Brent Mittelstadt, and Chris Russell. "Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR". In: *Harvard Journal of Law & Technology* 31.2 (2018), pp. 841–887.

[107] Michael Kearns and Aaron Roth. *The Ethical Algorithm: The Science of Socially Aware Algorithm Design*. New York: Oxford University Press, 2019.