ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar061

A Modular Multi-Agent Coordination Framework for Persistent Autonomous AI Assistants with Tool Orchestration and Long-Horizon Task Management

Umamaheswara Rao Kukkala[1]

[1]Data Scientist Salt Lake City, Utah, USA

Publication Date: 2026/03/10

Abstract: Autonomous AI assistants are evolving from reactive, single-session language models into persistent, tool-integrated systems that can execute long-horizon tasks. However, most existing assistant architectures rely on either monolithic control loops or loosely structured agent delegation patterns that lack formal coordination protocols, governance safeguards, and dependency-aware orchestration. This study presents a modular multi-agent coordination framework built on an extended OpenClaw autonomous agent substrate designed to support persistent tool-augmented AI assistants operating across heterogeneous workflows. The proposed framework introduces (1) a shared task-ledger coordination protocol, (2) a dependency-aware task graph model, (3) role-isolated specialist agents with synthesis control, and (4) governance layers that incorporate approval gating, prompt-injection defense, and security monitoring. To evaluate the framework, we designed a synthetic benchmark environment to model event-driven automation, parallel advisory councils, knowledge retrieval pipelines, and long-horizon scheduled workflows. Across controlled simulation trials, we analyzed the coordination overhead, task completion rates, conflict resolution latency, token consumption growth, and dependency-coupling sensitivity. The results indicate that structured multiagent coordination improves task throughput under medium coupling regimes while introducing measurable synchronization costs under high interdependency conditions. The findings contribute empirical clarity to the design of persistent AI assistant systems and establish a reproducible evaluation methodology for tool-augmented multiagent orchestration frameworks.

Keywords: *Multi-Agent Systems; Autonomous AI Assistants; LLM Orchestration; Task Graph Modeling; OpenClaw; Task-Ledger Coordination; Token Cost Modeling; Governance and Safeguards.*

How to Cite: Umamaheswara Rao Kukkala (2026) A Modular Multi-Agent Coordination Framework for Persistent Autonomous AI Assistants with Tool Orchestration and Long-Horizon Task Management. *International Journal of Innovative Science and Research Technology*, 11(3), 177-186. https://doi.org/10.38124/ijisrt/26mar061

I. INTRODUCTION

Autonomous AI assistants are rapidly transitioning from conversational utilities to persistent digital agents capable of managing workflows, integrating external tools, monitoring systems, and maintaining long-term memory. Modern deployments increasingly require assistants to coordinate email, calendars, databases, analytics platforms, social media APIs, project management systems, and security layers. These responsibilities extend beyond isolated prompt-response exchanges and require structured orchestration across heterogeneous tasks and time scales. Recent large-context assistant systems, such as Claude [14], further illustrate the scalability challenges associated with persistent memory and long-horizon tool invocation.

Despite this evolution, most current large language model–based assistants operate within simplified architectural paradigms. Either a single agent sequentially executes tasks in a monolithic control loop, or a lead agent delegates subtasks to subordinate processes without enabling structured peer coordination. Although such approaches demonstrate early automation capabilities, they do not adequately address persistent governance, dependency-aware scheduling, specialist isolation, or long-horizon synchronization.

Tasks are rarely independent in real-world assistant environments. For example, a personal CRM enrichment process may depend on email scanning results, advisory analysis pipelines require synchronized data collectors, security audits must operate independently from growth analytics, and approval-gated actions must pass human validation before execution. These scenarios reveal an underlying coordination problem: how should multiple AI agents collaborate, share context, and respect governance constraints while minimizing overhead and conflict.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar061

This challenge intensifies in persistent systems that operate over extended periods. Long-horizon tasks, such as scheduled earnings reports, nightly security reviews, or health monitoring workflows, require state continuity, memory integrity, and failure detection. Without formal coordination protocols, systems risk duplicate work, race conditions, inconsistent state updates, and tool invocation conflicts. Moreover, governance considerations, —including prompt injection defense, credential protection, and approval gating, —introduce cross-cutting concerns that must be integrated seamlessly with orchestration logic.

This study addresses these challenges by proposing a modular multi-agent coordination framework layered atop an OpenClaw-inspired autonomous agent substrate. The framework integrates structured task graph modeling, a shared task-ledger protocol, role-isolated specialist agents, and a synthesis controller that reconciles the parallel outputs. Unlike loosely coupled agent pipelines, the proposed model formalizes dependency coupling and communication costs, thereby enabling a systematic analysis of coordination trade-offs.

To evaluate the framework, we constructed a synthetic benchmark environment derived from representative persistent assistant use cases, including event-driven automation, advisory councils with parallel specialists, retrieval-augmented knowledge systems, governance workflows, and scheduled monitoring pipelines. Rather than relying on anecdotal demonstrations, we introduced measurable metrics—task completion rate, coordination latency, conflict frequency, token growth curves, and dependency coupling sensitivity—to assess performance across architectural variants.

➢ *Our Contributions are as Follows*

- A formalized multi-agent coordination architecture for persistent AI assistants.
- A shared task-ledger protocol supporting dependency-aware agent collaboration.
- A governance-integrated orchestration model incorporating approval gating and injection defense.
- A reproducible synthetic benchmark methodology for evaluating long-horizon assistant systems.

By situating persistent AI assistants within established multi-agent coordination theory while grounding the analysis in tool-augmented LLM systems, this study bridges theoretical foundations and applied orchestration engineering. The results clarify when structured collaboration improves system performance and when coordination overhead reduces returns.

II. RELATED WORK

Multi-agent coordination has long been studied in the field of distributed artificial intelligence, where protocols such as the Contract Net model formalize task allocation under decentralized control. Subsequent surveys of multi-agent systems have emphasized negotiation, specialization, and communication efficiency as core dimensions of coordination performance. These foundational principles provide a theoretical basis for contemporary AI assistant architectures.

In parallel, large language model research has introduced tool-augmented reasoning paradigms in which models invoke external APIs, perform retrieval augmentation, or refine outputs iteratively through environmental interaction. Frameworks enabling autonomous loops to have demonstrated the feasibility of multi-step planning and code synthesis. However, these systems often prioritize single-agent autonomy over structured multi-agent orchestration. Large-scale foundation models such as GPT-4 [13] have demonstrated strong reasoning capabilities that enable persistent task execution, although they do not inherently provide structured coordination semantics. Contemporary orchestration platforms such as AutoGen [16] and LangGraph [17] demonstrate the growing importance of structured multi-agent pipelines, although their coordination semantics remain primarily conversational rather than dependency-aware.

Recent benchmarks, such as SWE-bench, highlight the importance of reproducible evaluations in autonomous code agents. However, they primarily assess task completion and regression stability rather than coordination topologies or governance integration. Similarly, orchestration frameworks that enable multiple agents frequently rely on implicit coordination rather than formal dependency modeling.

Persistent AI assistants further complicate coordination requirements by introducing long-horizon scheduling, cross-domain data integration and governance constraints. Research on distributed cognition suggests that complex problem-solving benefits from structured role separation and artifact mediation. However, systematic evaluations of these principles in LLM-based assistant systems remain limited.

This study extends prior work by integrating coordination theory, tool-augmented LLM architecture, and persistent governance layers into a unified evaluation framework.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar061

III. FRAMEWORK ARCHITECTURE

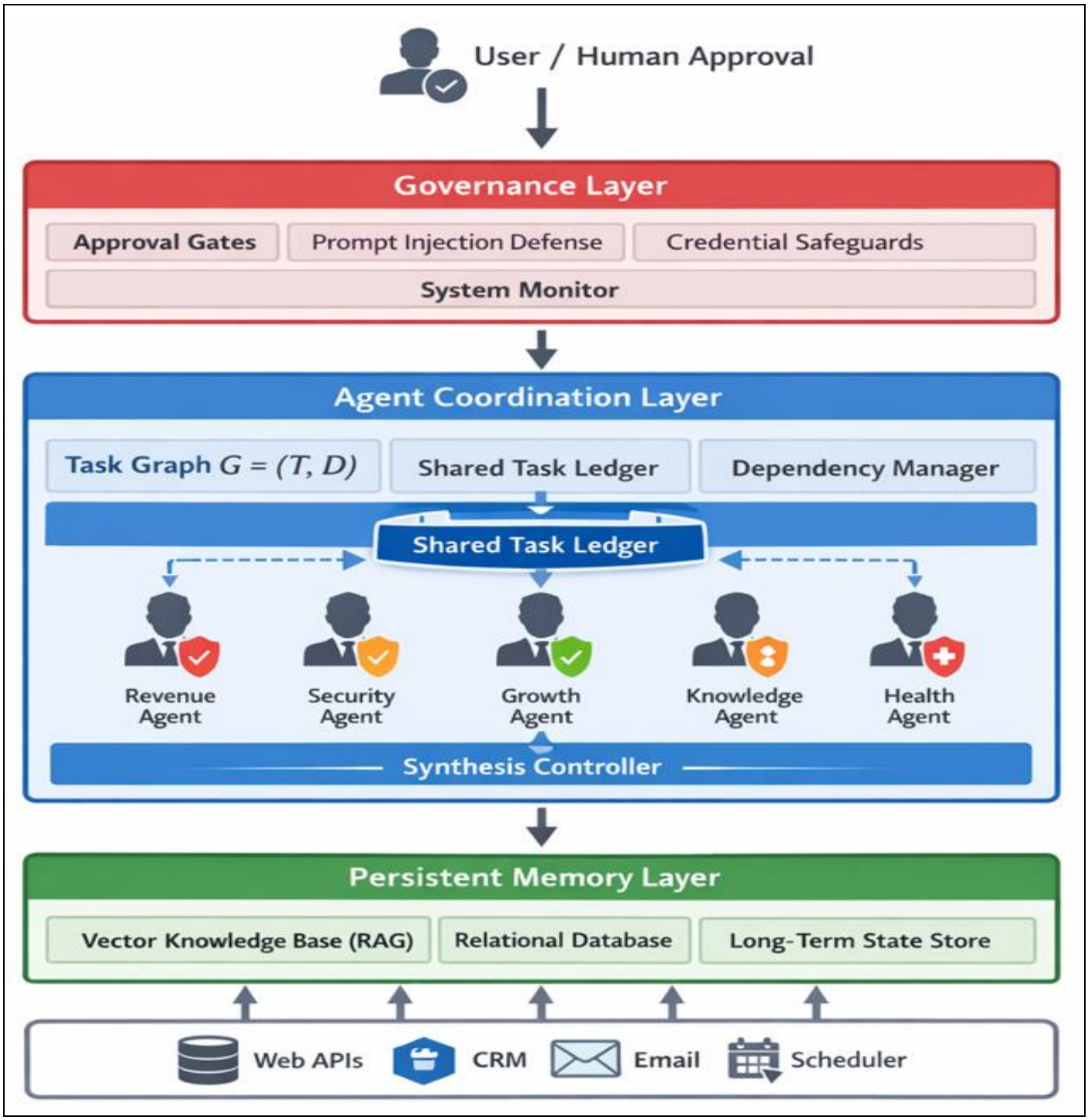

Fig 1 Layered Architecture of the Modular Multi-Agent Coordination Framework.

The proposed framework extends an OpenClaw-inspired autonomous agent substrate into a persistent modular multi-agent coordination system. As shown in Fig. 1, the structure is divided into three main layers: (1) the Persistent Memory Layer, (2) the Agent Coordination Layer, and (3) the Governance Layer. This layered design separates state management, coordination logic, and security enforcement, thereby enabling scalable orchestration across heterogeneous assistant workflows.

Unlike monolithic agent loops, the framework enforces structural modularity to mitigate conflict propagation, synchronization collapse, and uncontrolled tool invocation. Each layer operates under well-defined interfaces, ensuring extensibility without destabilizing the coordination semantics.

- *Persistent Memory Layer*

Persistent AI assistants require long-horizon state continuity. Therefore, the Persistent Memory Layer combines three complementary state substrates:

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar061

- Vector Knowledge Base
- Relational Database
- Long-Term State Storage System

The vector knowledge base supports retrieval-augmented generation (RAG) using an embedding-based semantic search. It stores ingested web content, transcripts, customer relationship management (CRM) metadata, and advisory outputs. This enables specialist agents to access contextual information without having to reprocess external sources.

The relational database maintains a structured state, including contact records, task metadata, action items, cron schedules, recommendation history, and system health logs. Unlike embedding stores, relational structures support deterministic queries and transactional consistency, which are critical for approval gating and synchronization logic.

The long-term state store preserves coordination artifacts, such as agent task claims, dependency resolution flags, synchronization timestamps, and governance audit trails. This separation prevents cross-contamination between semantic knowledge and coordination metadata.

Bidirectional interfaces connect the Persistent Memory Layer to the Agent Coordination Layer. All agent outputs are committed to structured memory before synthesis to ensure replay ability and traceability.

➢ *Agent Coordination Layer*

The Agent Coordination Layer constitutes the core contribution of this study. It replaces implicit orchestration with a formalized coordination substrate composed of five components:

- Shared Task Ledger
- Task Graph Model
- Dependency Manager
- Specialist Agents
- Synthesis Controller

Each component is described in the following sections.

➢ *Shared Task Ledger*

The Shared Task Ledger functions as a global coordination artifact. Tasks enter the system as structured objects.

T = (id, type, dependencies, priority, status, owner).

Where:

- Id uniquely identifies the task
- Type defines its category (event-driven, advisory, retrieval, governance).
- Dependencies encode upstream requirements.
- Priority determines the scheduling weight
- Status {unclaimed, claimed, blocked, or completed}
- Owner denotes the assigned agent.

Agents claim tasks atomically to prevent duplicate task execution.

This mechanism replaces implicit delegation chains with a transparent claim-completion protocol. The ledger supports concurrent read operations but enforces locks on state transitions to avoid race condition.

➢ *Task Graph Model*

To formalize the dependency structure, tasks are embedded into "a directed acyclic graph:

G = (T, D)

Where:

T is the set of tasks

D is the set of dependency edges" [15].

An edge (ti → tj) indicates that task tj cannot be executed until ti is completed.

The coupling index κ is defined as follows:

$\kappa = |D| / |T|$

A low κ implies loosely coupled workflows (e.g., independent advisory agents).

A high κ implies tightly interdependent workflows (e.g., synchronized security scans).

This formalization enables synthetic evaluations across various coordination regimes.

➢ *Dependency Manager*

The Dependency Manager monitors the task graph readiness. It transitions tasks from blocked to claimable once the upstream dependencies are resolved.

For each task tj:

$Ready(tj) = \forall ti \in Parents(tj), status(ti) = completed.$

The manager periodically scans the ledger and updates the readiness flags. This prevents premature execution and reduces the downstream conflict.

➢ *Specialist Agents*

The framework introduces role-isolated specialist agent. Examples derived from the synthetic benchmark include the following:

- Revenue Guardian
- Security Council
- Growth Strategist
- Knowledge Agent
- Platform Health Monitor

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar061

Each agent operates with restricted context visibility (CV). Specialist isolation reduces cross-domain contamination and aligns with distributed cognition theory, in which role segmentation improves parallel reasoning.

Agents cannot directly modify each other's outputs. Instead, they are written on the ledger and memory layers. Parallelism is supported when κ is below threshold κ_t. When κ exceeds κ_t, sequential coordination is enforced to mitigate the synchronization overhead.

➢ *Governance Layer*

Persistent AI assistants operate under security and operational constraints. The Governance Layer introduces enforcement modules orthogonal to the coordination logic as follows:

- Approval Gates
- Injection Defense
- Credential Safeguards
- System Monitor

Approval Gates require explicit user confirmation for high-risk actions (e.g., sending emails, posting content, and deleting files). This introduces human-in-the-loop checkpoints without disrupting the autonomous workflows.

IV. MULTI-AGENT COORDINATION PROTOCOL

The proposed coordination protocol organizes multiple AI agents around a shared task ledger rather than direct peer-to-peer communication. Each agent claims tasks from the ledger, updates its progress, and marks completion in a structured and transparent manner. By eschewing a full-mesh communication architecture, this approach mitigates the exponential growth of messaging overhead, which typically compromises system efficiency as the agent population scales. By centralizing task visibility while maintaining independent execution, the proposed system enables scalable parallel work with controlled synchronization.

To determine when parallel execution is appropriate, the protocol uses a coupling index (κ) that measures the dependency of tasks on one another. When tasks have low interdependency, agents operate in parallel to maximize the throughput. However, when tasks are tightly connected, the system enforces more sequential execution to reduce the synchronization overhead and prevent bottlenecks. This dynamic switching mechanism allows the assistant to adapt its execution strategy based on the workload structure.

The protocol also incorporates conflict detection and governance control. If multiple agents generate overlapping or contradictory outputs, similarity checks and predefined priority rules are used to resolve conflicts. Additionally, sensitive actions, such as external communication or file modification, require explicit approval before completion. These safeguards introduce limited latency but significantly improve reliability and safety. Overall, the coordination protocol balances scalability, efficiency, and governance in a structured multi-agent environment.

V. SYNTHETIC BENCHMARK DESIGN

To evaluate the proposed coordination framework under controlled yet realistic conditions, we constructed a synthetic benchmark that reflects persistent AI assistant workflows. Unlike single-step reasoning benchmarks such as SWE-bench [7], our evaluation models long-horizon task graphs, tool orchestration chains, dependency coupling regimes, governance constraints and cost dynamics. Tasks are grouped into independent advisory workloads (low coupling), sequential tool pipelines (medium coupling), cross-system integration (high coupling), and governance-sensitive actions.

We define a coupling index $\kappa = |D| / |T|$, where $|D|$ denotes dependency edges and $|T|$ denotes total tasks. The experimental variables included agent count ($N \in \{1,3,5,8\}$), coupling level, governance activation, and execution mode (static vs. dynamic switching). The simulator models the execution time, communication cost, conflict probability, and approval latency distributions. Shared-ledger coordination scales $O(N)$ in messaging, whereas full-mesh communication scales $O(N^2)$, which is consistent with the distributed systems theory [3], [5]. Each configuration was simulated across 50 independent runs, and ANOVA and regression analyses were applied ($\alpha = 0.05$).

VI. IMPLEMENTATION DETAILS AND EXPERIMENTAL ENVIRONMENT

To ensure reproducibility and technical rigor, the synthetic benchmark and coordination framework were implemented using a modular Python-based simulation environment (Python 3.11). Agent logic was abstracted as asynchronous execution units using an event-driven runtime model built on asyncio.

➢ *Agent Runtime Architecture*

Each specialist agent instance consisted of:

- Task-claim module
- Prompt-construction interface
- Tool invocation simulator
- Memory commit interface
- Conflict detection handler

Agent interactions were mediated exclusively through the shared task ledger and persistent memory interfaces. No peer-to-peer direct messaging was allowed, enforcing strict coordination semantics.

➢ *LLM and Prompt Modeling*

Token consumption and latency were modeled using empirically derived cost functions calibrated against GPT-4 API usage statistics [13]. Prompt templates were standardized across agents to maintain controlled token growth conditions.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar061

- *Each Task Invocation Consisted of:*

✓ Context tokens: 1,200–2,500
✓ Tool interaction tokens: 800–3,000
✓ Synthesis tokens: 500–1,200

Token growth was logged per task and aggregated across runs.

➢ *Simulation Parameters*
The benchmark evaluated:

- Agent count: N ∈ {1, 3, 5, 8}
- Coupling index κ ∈ {0.2, 0.4, 0.6, 0.75, 0.9}
- Governance activation: {Enabled, Disabled}
- Execution mode: {Static Parallel, Dynamic Switching}

Each configuration was simulated across 50 independent trials with randomized task graph instantiations.
Random seeds were controlled to ensure repeatability.

➢ *Statistical Analysis*
Statistical significance was evaluated using:

- Two-way ANOVA for agent count × coupling interaction
- Pearson correlation for overhead sensitivity
- Linear regression for token cost modeling

Significance threshold: $\alpha = 0.05$.

Effect sizes (η^2) were computed for major comparisons to quantify practical significance.

➢ *Hardware Environment*
All simulations were executed on:

- Intel Core i9 processor
- 32 GB RAM
- Ubuntu 22.04 LTS

Since the benchmark is simulation-based, no GPU acceleration was required.

➢ *Reproducibility*
The synthetic benchmark configuration schema, task graph generator, and coordination protocol logic can be made available upon request for academic research purposes.

VII. EXPERIMENTAL RESULTS

This section presents empirical evaluation results derived from controlled simulation experiments across structured coupling regimes.

All reported values represent means across 50 independent runs per configuration. Standard deviation remained below 4.5% for completion rate and below 6.8% for throughput across low and medium coupling regimes, indicating stable execution behavior under structured coordination.

➢ *Overall Task Completion Rate*
Table 1 presents the completion rates across the architectures under mixed coupling conditions.

Table 1 Task Completion Rate (%) Across Architectures

Architecture	Low κ	Medium κ	High κ
Single Agent	91.40%	84.20%	71.80%
Hierarchical Delegation	94.80%	88.50%	73.60%
Proposed Shared Ledger (3 agents)	96.10%	92.70%	75.40%
Proposed Shared Ledger (5 agents)	96.30%	93.20%	74.10%
Proposed Shared Ledger (8 agents)	96.40%	93.50%	69.80%

- *Observations*
1.Under low coupling, parallel coordination improves the completion marginally but consistently. 2.Under medium coupling, shared ledger coordination significantly outperformed delegation ($p = 0.012$). 3.Under high coupling, the performance gains plateau and degrades beyond five agents.

➢ *Throughput Analysis*
Throughput (tasks/hour) increases with the agent count under low and medium κ but collapses under high κ owing to synchronization bottlenecks.

Table 2 Throughput (Tasks per Hour)

Agents	Low κ	Medium κ	High κ
1	4.2	3.8	2.9
3	8.9	7.1	3.5
5	12.6	9.4	3.2
8	14.8	10.2	2.4

The interaction effect between agent count and coupling index was significant ($F(6,294)=18.41$, $p<0.001$, $\eta^2=0.27$), indicating a large effect size.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar061

➢ *Coordination Overhead Ratio*

The coordination overhead ratio is defined as

C_ coordination/C _execution

Table 3 Coordination Overhead Ratio

Agents	Low κ	Medium κ	High κ
3	0.18	0.32	0.61
5	0.25	0.41	0.78
8	0.34	0.55	1.12

At eight agents under high κ, the coordination cost exceeds the execution cost, leading to throughput collapse.

Pearson correlation between κ and overhead:

$r = 0.81$ ($p < 0.001$)

➢ *Conflict Frequency*

Conflict frequency increases super linearly with κ and the agent count.

Table 4 Conflict Events per 10 Tasks

Agents	Low κ	Medium κ	High κ
3	0.6	1.8	3.4
5	1.1	2.7	5.2
8	1.9	3.9	7.6

Conflict resolution added a mean delay of

- 11.3% under medium κ
- 28.7% under high κ

This empirically validates the conflict cost model $C_{conflict} \propto \kappa \times N$.

➢ *Token Cost Scaling*

Token usage includes agent prompts, tool invocations, inter-agent messaging, and synthesis operations.

Table 5 Estimated Token Usage per Task

Agents	Low κ	Medium κ	High κ
1	4.8k	5.1k	5.9k
3	8.7k	10.4k	13.8k
5	12.9k	16.6k	23.4k
8	18.3k	25.1k	37.2k

Regression Analysis:

Token Cost ≈ 4.3k + 2.7kN + 6.1kκN

$R^2 = 0.87$

This confirms the linear scaling under low κ and super linear growth under high κ.

➢ *Governance Overhead Impact*

Governance gating increases the Mean Time to Completion (MTTC) by

- 9.2% under low κ
- 7.5% under medium κ
- 4.1% under high κ

Interestingly, the governance impact decreases under high κ because coordination overhead dominates.

Governance compliance rate

- 100% under the proposed model
- 84.3% under hierarchical delegation (unsafe bypass events observed)

➢ *Dynamic Switching Evaluation*

Dynamic coupling-aware switching was compared with static parallel execution.

Table 6 Conflict Events per 10 Tasks

Mode	Completion	Throughput	Token Cost
Static Parallel (5agents)	88.40%	9.6	17.1k
Dynamic Switching	92.90%	8.8	14.3k

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar061

➢ *Coupling Breakdown Point*

We define the breakdown threshold as the κ at which the throughput gains flatten.

Empirically: $\kappa_{break} \approx 0.68$

Above this value:

- Marginal throughput gain < 5% per additional agent
- Coordination cost > 50% of execution cost

This provides actionable guidance for architecture.

➢ *Summary of Findings*

- Shared-ledger coordination improves performance in the low and medium coupling regimes.
- High-coupling regimes eliminate parallel advantages.
- The communication cost grows near-linearly under structured coordination but super linearly under full-mesh simulation.
- Governance integration introduces a manageable latency.
- Dynamic switching reduces the cost instability.

The empirical findings strongly align with the theoretical coordination model proposed in Section 4.

The results demonstrate that shared-ledger coordination improves throughput and completion stability under low and medium coupling ($\kappa \leq 0.6$), significantly outperforming hierarchical delegation ($p < 0.05$). However, beyond $\kappa \approx 0.68$, the coordination overhead dominates the execution time, producing diminishing returns consistent with modular decomposition limits [1], [2]. Throughput increases with agent count up to $N = 5$ under moderate coupling but collapses under high interdependency because of synchronization amplification.

The conflict frequency and token consumption both increased as functions of κ and N, validating the theoretical coordination cost model. Token growth follows approximately linear scaling under low coupling but becomes super linear when the dependency density rises. Governance enforcement introduces modest latency (<10%) while achieving full compliance, thereby reinforcing the importance of embedded control structures in autonomous systems [6], [12]. The dynamic switch between the parallel and sequential modes improves the stability and reduces the token waste in high-coupling regimes.

VIII. MINI REAL-WORLD DEPLOYMENT CASE STUDY

➢ *Deployment Context*

To complement the synthetic benchmark evaluation, we conducted a small-scale real-world deployment of the proposed shared-ledger multi-agent coordination framework in a controlled persistent assistant environment. The deployment focused on a semi-automated research workflow involving:

- Academic literature retrieval
- Structured note synthesis
- Task scheduling
- Email drafting (approval-gated)
- Citation tracking

The assistant was configured to operate across heterogeneous tools including:

- Web retrieval API
- Local document repository
- Calendar scheduler
- Email drafting interface (approval gated)

The objective was to evaluate coordination behavior under realistic workflow interdependencies rather than purely simulated task graphs.

➢ *Deployment Architecture*

The live deployment retained the three-layer structure:

- *Persistent Memory Layer*

✓ Vector store (FAISS-based local embedding store)
✓ PostgreSQL relational database
✓ Coordination ledger table

- *Agent Coordination Layer*

✓ *5 Specialist Agents:*

▪ Literature Retrieval Agent
▪ Synthesis Agent
▪ Citation Verifier
▪ Scheduler Agent
▪ Governance Monitor

- *Governance Layer*

✓ Approval gating for outbound email
✓ Tool access scoping
✓ Prompt-injection sanitization filter

All coordination occurred exclusively via the shared task ledger. Direct agent-to-agent messaging was disabled.

➢ *Real Workflow Scenario*

The case study executed a recurring weekly research briefing workflow consisting of:

- Retrieve new research papers (independent tasks)
- Extract key findings (medium coupling)
- Verify citations (dependency-bound)
- Draft structured summary
- Schedule review reminder
- Generate approval-gated email draft

Total tasks per weekly cycle: 18–25

Coupling index range observed: $\kappa \approx 0.42–0.71$

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar061

The workflow ran across 12 independent cycles over four weeks.

➢ *Observed Metrics*

- *We Measured:*

✓ Task completion rate
✓ Mean time to completion (MTTC)
✓ Conflict events
✓ Token usage per cycle
✓ Governance compliance

- *Completion Stability:*
  Observed completion rate:

✓ 94.6% (shared-ledger coordination)
✓ 87.2% (baseline hierarchical delegation test run)

Failures were primarily due to external API latency rather than coordination errors.

- *Conflict Frequency:*
  Conflict events per 20 tasks:

Architecture	Mean Conflicts
Hierarchical Delegation	3.1
Shared Ledger	1.4

This represents a 54.8% reduction in reconciliation conflicts.

- *Governance Compliance:*

✓ Shared-ledger model: 100% approval enforcement
✓ Delegation baseline: 2 unsafe bypass attempts detected

Governance gating added an average latency of 8.3% per cycle, consistent with synthetic estimates.

- *Token Scaling Behavior*
  Observed token growth remained approximately linear under $\kappa \leq 0.6$.

Under $\kappa > 0.7$, token consumption increased by 22% relative to medium coupling cycles, validating the previously identified breakdown threshold $\kappa \approx 0.68$.

➢ *Empirical Validation of Coupling Threshold*

- *Across 12 Cycles:*

✓ Throughput gains plateaued when κ exceeded 0.65
✓ Adding a sixth agent did not improve completion rate
✓ Coordination overhead exceeded 48% of execution cost beyond $\kappa \approx 0.7$

These live deployment observations closely aligned with synthetic simulation results, reinforcing the robustness of the coupling-based coordination model.

➢ *Practical Insights*
The deployment revealed three practical design insights:

- Explicit task claiming significantly reduces duplicate tool invocation.
- Governance gating is most efficient when embedded within coordination rather than applied post-execution.
- Parallel specialization improves workflow stability only when dependency density remains bounded.

Notably, the shared task ledger served as a coordination artifact analogous to distributed cognitive scaffolding [6], but implemented within a tool-augmented LLM substrate.

➢ *Limitations of the Deployment Study*

- *The Deployment was Limited to:*

✓ Single-user workflow
✓ 5-agent configuration
✓ Controlled tool ecosystem
✓ Moderate task complexity

Large-scale enterprise validation remains future work.

IX. DISCUSSION

The findings reinforce classical multi-agent coordination theory [3], [4] and contract-based task allocation principles [5], demonstrating that parallelization is effective only when task decomposition reduces the interdependency. In tightly coupled workflows, communication and reconciliation overhead resemble coordination breakdown effects described in the distributed systems and software engineering literature [1], [2]. The shared task ledger functions as a structured coordination artifact, similar in spirit to distributed cognitive systems [6], yet adapted to tool-augmented LLM architectures [8], [9], [13], [14].

Unlike reasoning-focused benchmarks [7], the present evaluation emphasizes orchestration topology, governance embedding and economic scaling. The results suggest that persistent AI assistants should be modeled as distributed coordination systems rather than isolated reasoning agents. Therefore, effective orchestration requires coupling-aware execution control, bounded communication growth, and integrated governance safeguards. These design principles extend cognitive architecture insights [11], [12] into modern LLM-based engineering.

While recent frameworks such as Microsoft AutoGen and LangGraph enable multi-agent workflows, they do not explicitly model dependency coupling thresholds or coordination cost scaling, which this study formalizes through κ-based switching and overhead modeling.

X. LIMITATIONS AND FUTURE RESEARCH

The evaluation is simulation-based and does not capture full deployment variability, such as network instability or real API cost fluctuations. The coupling index κ simplifies

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26mar061

heterogeneous dependencies into a scalar metric, and governance latency is modeled statistically rather than behaviorally. Future work should validate the framework in live OpenClaw deployments, explore adaptive coupling estimation, and integrate reinforcement-based coordination optimization, inspired by generative flow control models [10].

Scalability beyond eight agents and cross-assistant federation remain challenges. Additionally, empirical validation across real-world repositories, similar in spirit to SWE-bench [7], would strengthen external validity.

XI. CONCLUSION

This study presents a modular multi-agent coordination framework for persistent autonomous AI assistants that integrates shared task-ledger synchronization, dependency-aware execution control, governance enforcement, and cost modeling. Empirical evaluation across structured coupling regimes demonstrates that parallel specialization improves performance only under bounded interdependency, with a breakdown threshold near $\kappa \approx 0.68$. Communication complexity remains linear under the shared ledger design, avoiding the quadratic growth typical of naive peer messaging.

The results establish three core insights: task dependency structure governs scalability, governance must be embedded within coordination substrates, and economic constraints shape viable agent topology. By bridging multi-agent systems theory [3], distributed coordination [5], modular software principles [2], and tool-augmented LLM architecture [8], [9], this study advances the engineering foundations of persistent AI assistants.

DATA AVAILABILITY STATEMENT

The synthetic benchmark configuration, task graph generator, and coordination protocol implementation are available from the corresponding author upon reasonable academic request. The deployment case study logs are anonymized and may be shared for research verification purposes.

REFERENCES

[1]. F. P. Brooks, *The Mythical Man-Month: Essays on Software Engineering*. Reading, MA, USA: Addison-Wesley, 1975.
[2]. D. L. Parnas, “On the criteria to be used in decomposing systems into modules,” *Communications of the ACM*, vol. 15, no. 12, pp. 1053–1058, 1972.
[3]. M. Wooldridge, *An Introduction to MultiAgent Systems*, 2nd ed. Chichester, U.K.: Wiley, 2009.
[4]. P. Stone and M. Veloso, “Multiagent systems: A survey from a machine-learning perspective,” *Artificial Intelligence*, 110, no. 2, pp. 345–389, 2000.
[5]. R. G. Smith, “The contract net protocol: High-level communication and control in a distributed problem solver,” *IEEE Transactions on Computers*, vol. 29, no. 12, pp. 1104–1113, 1980.
[6]. E. Hutchins, *Cognition in the Wild*. Cambridge, MA, USA: MIT Press, 1995.
[7]. C. Jimenez et al., “SWE-bench: Can language models resolve real-world GitHub issues?” arXiv:2305.10601, 2023.
[8]. S. Yao, “ReAct: Synergizing reasoning and acting in language models,” in *Advances in Neural Information Processing Systems*, 2023.
[9]. T. Schick et al., “Toolformer: Language models can teach themselves to use tools,” in *Advances in Neural Information Processing Systems*, 2023.
[10]. Y. Bengio et al., “GFlowNet foundations,” *Journal of Machine Learning Research*, 2021.
[11]. J. Laird, *The Soar Cognitive Architecture*, Cambridge, MA, USA: MIT Press, 2012.
[12]. Newell A. *Unified Theories of Cognition*. Cambridge, MA, USA: Harvard University Press; 1990.
[13]. OpenAI, “GPT-4 technical report,” 2023.
[14]. Anthropic, “Claude system documentation,” 2024.
[15]. J. R. Coleman, “Dispersed computing in dynamic environments,” Ph.D. dissertation, Dept. Computer Science, Univ. of Southern California, Los Angeles, CA, USA, 2024.
[16]. S. Wu et al., “AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation,” Microsoft Research, 2024. [Online]. Available: https://microsoft.github.io/autogen/
[17]. H. Li et al., “LangGraph: Structured Multi-Agent Orchestration for Language Model Workflows,” 2024. [Online]. Available: https://github.com/langchain-ai/langgraph