N. L. Kisialeu

MULTI-AGENT SYSTEM WITH DYNAMIC ROUTING AND TOOL-AWARE EXECUTION FOR ENTERPRISE WORKFLOW AUTOMATION

This paper presents the architecture of a production-grade multi-agent system for enterprise workflow automation. The system employs supervisor-based orchestration built on LangGraph, hybrid graph-semantic retrieval through Neo4j and ChromaDB, Model Context Protocol for tool integration, and a Human-in-the-Loop service. Evaluation over a four-month deployment demonstrates 35% improvement in answer relevance, 37% reduction in hallucination rate, and 28% decrease in token consumption.

Introduction

Large language models have demonstrated remarkable capabilities in natural language understanding, yet their enterprise application presents challenges: deterministic behavior, auditable decision trails, and integration with heterogeneous data sources [1]. Single-agent architectures struggle as task complexity grows, leading to context window exhaustion and degraded response quality.

Multi-agent architectures address this by decomposing tasks into specialized subtasks handled by dedicated agents [2]. Retrieval-Augmented Generation (RAG) grounds LLM responses in external knowledge [3], but standard pipelines rely on vector search alone, missing structured entity relationships. Hybrid retrieval combining graph-based and vector-based search captures both relational and semantic information [4].

This work presents a multi-agent platform integrating supervisor-based dynamic routing, hybrid graph-semantic retrieval, and extensible tool interfaces via the Model Context Protocol [5] for enterprise workflow automation.

I. System architecture and orchestration

The platform follows Clean Architecture with four layers: domain, application, infrastructure, and interfaces. The orchestration layer implements a supervisor-based multi-agent pattern using LangGraph [6]. The supervisor dynamically routes queries to specialized agents based on required domain expertise, expected tool interactions, and conversation history.

The agent registry maintains specialized agents for HR operations, document processing, knowledge retrieval, candidate matching, meeting coordination, and general conversation. Each agent is configured with domain-specific system prompts, tool bindings, and behavioral constraints. State management across multi-turn interactions uses LangGraph checkpointing backed by Redis, preserving context and tool results across agent transitions.

The knowledge layer combines Neo4j graph database for structured entity relationships (employees, skills, projects) with ChromaDB vector store for semantic search over unstructured content. A hybrid retrieval mechanism first executes skill-matching queries in Neo4j, then performs semantic search in ChromaDB, and merges results via reciprocal rank fusion.

The RAG pipeline accepts documents in PDF, DOCX, Markdown, and code formats. A `SplitterFactory` selects chunking strategies by file type: `RecursiveCharacterTextSplitter` for plain text, `MarkdownHeaderTextSplitter` for structured documents, and language-specific splitters for source code. Binary files are processed through a vision LLM that generates textual descriptions. Each chunk is enriched with metadata enabling fine-grained filtering and provenance tracking.

The tool layer implements the Model Context Protocol (MCP) [5], an open standard for typed tool discovery and invocation at runtime. MCP servers expose capabilities as typed descriptors that agents discover dynamically, decoupling agent logic from service implementations. Each agent accesses only domain-relevant tools through the dependency injection container.

The supervisor implements a two-phase routing strategy: classification of the incoming query by required expertise and complexity, followed by dispatch to the target agent with relevant context. Prompt engineering ensures reliability through structured system prompts specifying roles, constraints, output formats, and escalation protocols. The Human-in-the-Loop service pauses execution at high-stakes decision points, presenting context to a human operator for approval.

The deployment follows a containerized microservices pattern via Docker Compose, comprising FastAPI backend, Neo4j, ChromaDB, Redis, MongoDB, and MinIO. Langfuse provides observability through complete traces of agent reasoning chains, tool invocations, and latency measurements.

II. Experimental evaluation

Evaluation was conducted over a four-month production deployment using three complementary methodologies: automated assessment via LLM-as-a-Judge with GPT-4o, component-level ablation studies, and infrastructure performance profiling.

Hybrid retrieval improved answer relevance from 0.62 (vector-only baseline) to 0.84 with the combined graph-semantic approach, representing a 35% gain. The graph component contributed structured relational context that pure semantic search could not capture, particularly for multi-hop queries requiring traversal of employee-skill-project relationships in Neo4j.

Hallucination rate, measured as the proportion of responses containing factually unsupported claims,

decreased by 37% compared to the baseline. This reduction is attributed to the RAG pipeline's metadata filtering mechanism, which restricts retrieval scope to verified document categories, and the source attribution system that requires each generated claim to reference a specific retrieved passage.

An ablation study quantified individual component contributions. Removing the graph-based retrieval channel reduced answer relevance by 18%. Disabling metadata filtering increased hallucination rate by 23%. Removing the HITL service did not affect automated metrics but reduced user-reported trust scores by 31%, confirming its importance for perceived system reliability.

Token consumption was reduced by 28% through prompt compression and routing optimization. The context compression module removes redundant passages from retrieved chunks before prompt injection, while the routing mechanism prevents unnecessary context forwarding between agents, ensuring each agent receives only relevant information.

Routing accuracy reached 91.3% across all query categories, with the fallback mechanism successfully recovering 78% of initially misrouted queries. Misrouting was most common for ambiguous cross-domain requests spanning both HR policy interpretation and document generation. End-to-end latency averaged 3.2 s for single-agent and 7.8 s for multi-agent workflows, with LLM inference contributing 62%, retrieval operations 21%, and tool execution 17%. Redis-backed state persistence introduced negligible overhead at less than 15 ms per checkpoint. The system sustained 47 concurrent user sessions during peak hours with no degradation in response quality, and load testing validated linear throughput scaling up to 200 concurrent sessions.

Conclusion

A production-grade multi-agent platform for enterprise workflow automation has been designed, implemented, and evaluated. The architecture integrates supervisor-based dynamic routing with LangGraph state management, hybrid graph-semantic retrieval through Neo4j and ChromaDB, MCP-based tool integration, and a HITL service for human oversight. The four-month production deployment confirms that hybrid retrieval significantly improves answer relevance (+35%) and reduces hallucination rates (-37%) compared to vector-only baselines, while modular containerized deployment enables independent scaling of individual components. Future work includes adaptive routing via reinforcement learning from user feedback, temporal graph reasoning for time-sensitive queries, and evaluation of fine-tuned smaller models for domain-specific agent roles.

References

1. Zhao, W. A Survey of Large Language Models / W. Zhao [et al.] // arXiv preprint arXiv:2303.18223. – 2023.
2. Wu, Q. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation / Q. Wu [et al.] // arXiv preprint arXiv:2308.08155. – 2023.
3. Lewis, P. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks / P. Lewis [et al.] // Advances in Neural Information Processing Systems. – 2020. – Vol. 33. – P. 9459–9474.
4. Gao, Y. Retrieval-Augmented Generation for Large Language Models: A Survey / Y. Gao [et al.] // arXiv preprint arXiv:2312.10997. – 2024.
5. Anthropic. Model Context Protocol Specification [Electronic resource]. – Mode of access: https://modelcontextprotocol.io. – Date of access: 20.03.2026.
6. LangGraph: Build Stateful, Multi-Actor Applications with LLMs [Electronic resource]. – Mode of access: https://github.com/langchain-ai/langgraph. – Date of access: 20.03.2026.

Kisialeu Nikita Leonidovich, student at the Faculty of Applied Mathematics and Computer Science, Belarusian State University, nkisialeu@gmail.com.

Scientific supervisor: Shaltaniuk Stanislav Vitalievich, senior lecturer at the Department of Mathematical Modeling and Data Analysis, Belarusian State University, sholtanyuk@gmail.com.