DOI 10.70286/ISU-15.04.2026.021

SEMANTIC KERNEL USAGE FOR ORCHESTRATION OF MULTI-AGENT LLM-BASED SYSTEMS TO SOLVE THE TASKS WHICH REQUIRE DYNAMIC INVOLVEMENT OF NEW AGENTS

Kutsan Vitalii
Master student
Lyashkevych Vasyl
Candidate of Technical Sciences, PhD, docent
Associate Professor
Department of System Design
Ivan Franko National University of Lviv, Ukraine

Relevance of the topic. The development of modern artificial intelligence (AI) systems is accompanied by an architectural shift: from monolithic architectures that try to represent all functions using a single large language model (LLM), to multi-agent systems (MASs) based on LLMs. The reason for this shift is that having a single agent perform too many roles, tools, and functions leads to distraction, incorrect tool selection, and reduced response quality. Related research shows that MASs allow for the distribution of roles, memory, tools, and responsibilities among specialized agents, and can therefore be considered a distinct approach to intelligent systems development [1-2].

For tasks whose subtasks cannot be fully defined in advance, an “orchestrator-workers” approach is promising: a central agent breaks down the task, delegates it to a designated executor, and summarizes the results. Anthropic recommends this approach for complex programming and research scenarios, as additional agents or new search branches may be required during execution [3-4].

In the context of computer science, the problem is not only in coordinating agents, but also in defining the boundaries of their functional suitability. For a number of specialized tasks, in particular in software engineering, standard LLMs without additional fine-tuning may not be reliable enough. Therefore, the decomposition model should include decision rules for which subtasks can be transferred to agents that require specialized models, and which should be left under human control or a hybrid execution loop [5].

In the Microsoft Semantic Kernel Framework, agent orchestration has been outsourced to a separate framework-level mechanism that supports multiple coordination modes and can adapt to specific scenarios. Microsoft documentation emphasizes that these features are still experimental. Therefore, comparing orchestration strategies in the Semantic Kernel is both an application-oriented and scientifically relevant task [6-8].

Problem statement. For complex tasks in computer science, particularly in software development, intellectual data analysis and knowledge management, a single LLM agent is often insufficient due to problems such as context overloading, role confusion, unstable tool selection, and weak process control. Although the transition to a MAS architecture can partially solve this problem, some questions of the most effective decomposition and orchestration strategies remain unresolved for tasks that require iterative improvement and dynamic involvement of new agents [4-5, 9]. Also, it is not always possible to predict from an engineering perspective which agents are needed, how the task should be decomposed, and what interaction routes between agents will arise during the execution process.

In open-ended problems, the structure of subtasks evolves dynamically, depending on intermediate results, identified uncertainties, new information needs, and analysis or synthesis of answers. In this context, it is not enough to simply design a static set of agents and their rigid interaction processes. We need a scientific approach to modeling the problem processing process to formally describe: decomposition conditions, rules for generating new agents, communication mechanisms between agents, constraints on coordination paths, and process closure criteria.

Therefore, the research problem is not only to choose between sequential orchestration and group chat orchestration, but also to design an architectural solution that can dynamically assemble agents in a simple way, adaptively rebuild execution, and explore system behavior under incomplete uncertainty. Given the multitude of possible execution paths, decomposition methods, and tool combinations, it is practically impossible to fully compute all scenarios, explore all agent configurations, and thoroughly test the concept in advance. Thus, we need not only an application solution, but also a research tool to conduct further experiments, adjust planning rules, and analyze cost, quality, and stability of execution.

Considering all of the above, the aim of this work is to develop an architecture and conceptual approach for coordinating multi-agent LLM systems based on a semantic core, which can then be used as a research platform to study agent coordination strategies, runtime decomposition mechanisms, and conditions for efficient and dynamic recruitment of new specialized agents.

The purpose of the research is to propose an architectural solution for orchestrating an LLM-based MAS based on the Semantic Kernel for tasks that require dynamic decomposition and the involvement of new agents, as well as to form the basis for further experimental research into the effectiveness of such an approach in comparison with sequential orchestration and group chat orchestration.

The object of research is the processes of coordination, decomposition and task execution in LLM-based MASs.

The subject of research is methods, models and architectural strategies for agent orchestration in the Semantic Kernel, in particular sequential orchestration, group chat orchestration and runtime decomposition mechanisms with dynamic formation and connection of specialized agents.

The study hypothesizes that for tasks with unpredictable subtask structures, a dynamically proxies-based architecture offers higher solution quality, greater

adaptability, and is more suitable for studying complex scenarios compared to a rigid sequential pipeline, despite its higher coordination overhead. On the other hand, for linear deterministic scenarios, sequential orchestration remains more economically feasible. Furthermore, the proposed architecture not only has practical application value but also significant value as a research tool, enabling the study of processes such as agent generation, task decomposition and execution strategy selection.

Research results and discussion. To validate the proposed hypotheses, we recommend focusing on fundamental methods of agent orchestration and comparing them based on several conceptually important characteristics. First, we discuss task decomposition types, the possibility of dynamically introducing new agents, process controllability levels, expected solution quality, coordination overhead, and the applicability of this approach to complex scenarios.

Two fundamental patterns stand out in the Semantic Kernel: sequential orchestration and group chat orchestration. In sequential orchestration, agents are organized into a pipeline, where each agent processes the output of the previous agent. This approach is suitable for multi-stage inference, document validation, and data processing pipelines because it offers simple flow control, low overhead, and high determinism [7].

In contrast, group chat orchestration simulates collaborative discussions between agents in a controlled environment, where a chat manager or selection strategy determines who should respond next. This model is better suited for collaborative and iterative problem solving, as they require continuous improvement, validation of intermediate results, and, if necessary, integration of other specialized agents [8].

Empirical studies have shown that the orchestration architecture directly affects the quality and stability of the results. For example, MyAntFarm.ai achieved 100% recommendation execution rate when developing a multi-agent orchestration for event response, while the baseline single-agent model achieved only 1.7%. The specificity and correctness of the recommendations were also significantly improved. Furthermore, the authors emphasized that this advantage is not due to speed, but rather to a more meaningful role structure and more robust quality assurance [9].

However, recent comparative studies show that the choice of framework architecture itself can have a fundamental impact on latency, throughput, planning accuracy, and coordination success rate. The study “Understanding Multi-Agent LLM Frameworks: Unified Benchmarks and Experimental Analysis” demonstrates that the choice of framework design can increase latency by more than a hundredfold and significantly reduce planning and coordination performance [10].

Another research approach focuses not only on correctness but also on the efficiency of the agent framework. The AgentRace test emphasizes the importance of evaluating the execution time, scalability, communication overhead, and tool invocation latency of LLM agent systems in real deployment scenarios [11]. This means that the comparison of orchestration strategies should be based on several standards.

Additionally, benchmarks of financial document processing applications show that different orchestration patterns exhibit different trade-offs between cost and

accuracy. For sequential pipeline, parallel fan-out with merge, hierarchical supervisor-worker and reflexive self-correcting loop, we compared F1 scores, document accuracy, latency, cost per document, and token efficiency. Therefore, in the context of Semantic Kernel, the main focus is not on comparing abstract "better" or "worse" methods, but rather on applying specific orchestration strategies to specific categories of tasks, in particular those that require dynamic decomposition and agent search [11].

Thus, the literature review shows that there is already a solid theoretical basis for LLM-based MASs, industry recommendations from Microsoft and Anthropic are available, and the first system benchmark studies are emerging. However, a notable gap remains: there is a lack of applied work that systematically compares orchestration patterns in the Semantic Kernel specifically for tasks with dynamic recruitment of new agents and evaluates them not only in terms of result quality, but also in terms of coordination costs, token costs, latency, and robustness of results. A generalized comparison of sequential orchestration, group chat orchestration, and the proposed dynamic agent orchestration approach is given in Table 1. This form of presentation allows not only to systematize existing approaches, but also to substantiate the feasibility of developing an architecture focused on runtime decomposition of the task and dynamic formation of specialized agents.

Table 1. A general comparison of orchestration approaches

Approach to orchestration	Type of decomposition	Dynamic involvement of new agents	Typical scenarios
Sequential orchestration	Predetermined, linear	No / limited	ETL, sequential document processing, fixed pipelines
Group chat orchestration	Partly dynamic	Limited, via interaction manager	Discussion, review, joint clarification of decisions
Dynamic agent orchestration (proposed approach)	Runtime decomposition	Yes	Research tasks, complex analytics, multi-step problem solving

Table 1 presents a conceptual comparison of three orchestration approaches for LLM-based MASs. It shows that the approaches differ mainly in how task decomposition is organized and whether new agents can be involved during their execution. Sequential orchestration is the most rigid approach: tasks are pre-broken into a fixed series of linear stages, making it best suited for deterministic processes such as ETL or sequential file processing. Group chat orchestration is more flexible because the interactions between agents can change during execution, but the addition of new agents remains limited and is usually controlled by the chat or interaction manager. This makes it suitable for collaborative scenarios such as iterative refinement of discussions, reviews, or decisions.

The proposed dynamic agent orchestration is the most adaptive approach. A high-level architecture of the proposed approach is shown in Figure 1. It supports decomposition at runtime, for example, the structure of subtasks can emerge during execution, and new specialized agents can be dynamically added as needed. Therefore,

this approach is most suitable for research tasks, complex analytics, and multi-step problem solving, where the sequence of actions cannot always be predicted in advance.

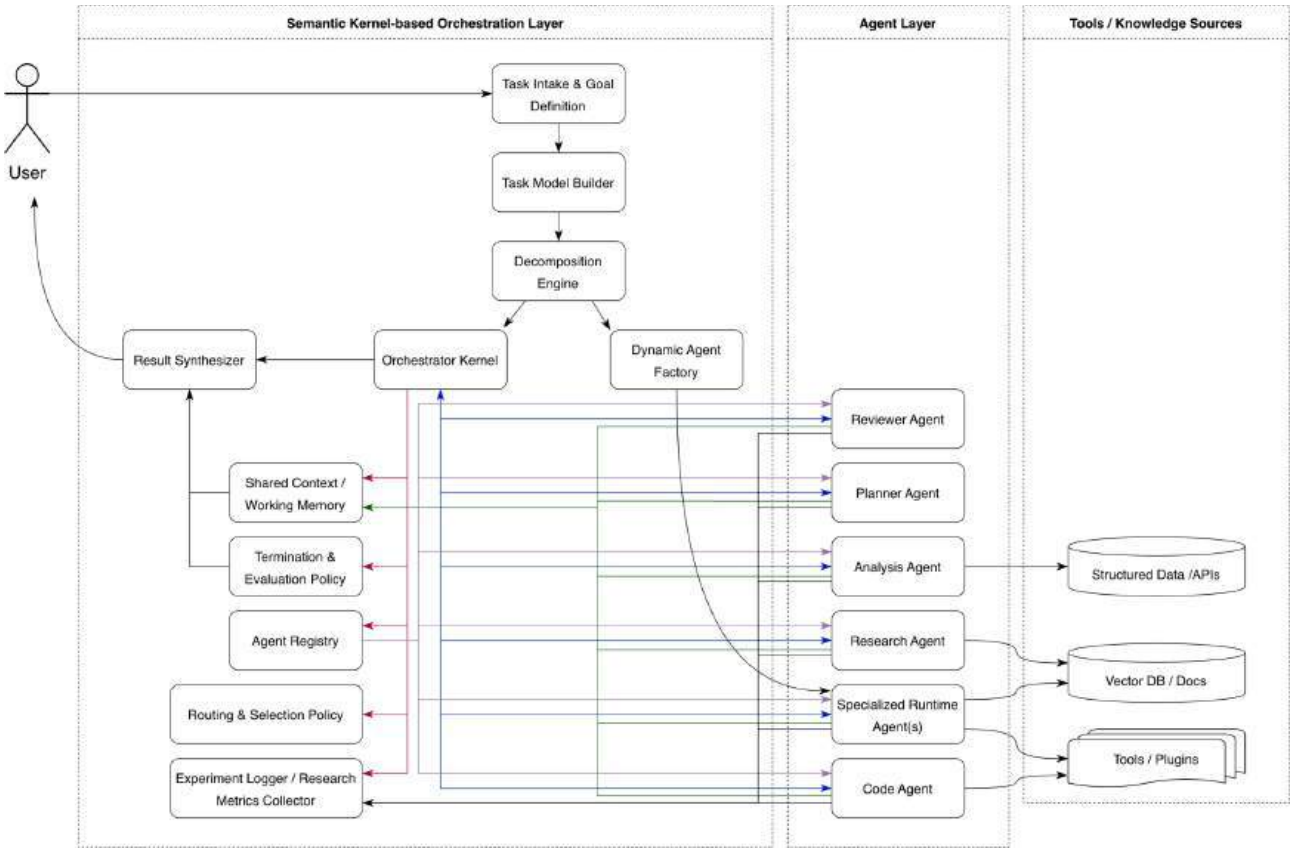

Fig. 1. A high-level architecture of the proposed orchestration approach.

In the proposed architecture, the task is first formalized as an object of research: the goal, constraints, uncertainties, and completion criteria are defined. Then, the Task Model Builder and Decomposition Engine do not simply transfer the task to agents, but form a model of its possible execution. If the existing set of agents is not enough, the Dynamic Agent Factory allows you to generate new specialized agents for runtime needs. The Orchestrator Core coordinates execution, relying on routing, completion, and evaluation policies. In parallel, the Experiment Logger / Research Metrics Collector collects data for future scientific analysis. This is what makes the system not only an execution mechanism, but also a research tool.

Conclusion. MASs with LLM-based agents using Semantic Kernel looks as a promising approach for solving tasks that require decomposition, coordination, and dynamic recruitment of new agents. Sequential orchestration is suitable for linear and well-structured processes, while group chat orchestration and, in general, selector-based methods are better suited for tasks with nonlinear structures that require improved intermediate results and adaptive planning. Additional research should aim to experimentally investigate the proposed approach in the Semantic Kernel across a set of metrics of quality, latency, token cost and infrastructure costs.

References

1. Li, X., Wang, S., Zeng, S., Wu, Y., & Yang, Y. (2024). A survey on LLM-based multi-agent systems: Workflow, infrastructure, and challenges. Vicinagearth, 1, Article 9. https://doi.org/10.1007/s44336-024-00009-2.
2. Chen, S., Liu, Y., Han, W., Zhang, W., & Liu, T. (2024). A survey on LLM-based multi-agent system: Recent advances and new frontiers in application. arXiv. https://arxiv.org/abs/2412.17481.
3. Anthropic. (2024, December 19). Building effective AI agents. Retrieved from: https://www.anthropic.com/engineering/building-effective-agents.
4. Anthropic. (2025, June 13). How we built our multi-agent research system. Retrieved from: https://www.anthropic.com/engineering/multi-agent-research-system.
5. Grynets, O., Lyashkevych, V., Baran, D., Orliansky, M., Zelenyy, T., & Leshchyshyn, M. (2025). Fine-tuned LLM-based Code Migration Framework. arXiv preprint arXiv:2512.13515. https://arxiv.org/abs/2512.13515.
6. Microsoft. (2025, July 21). Semantic Kernel agent orchestration. Microsoft Learn. Retrieved from: https://learn.microsoft.com/en-us/semantic-kernel/frameworks/agent/agent-orchestration/.
7. Microsoft. (2025, July 21). Sequential agent orchestration. Microsoft Learn. Retrieved from: https://learn.microsoft.com/en-us/semantic-kernel/frameworks/agent/agent-orchestration/sequential.
8. Microsoft. (2025, May 22). Group chat agent orchestration. Microsoft Learn. Retrieved from: https://learn.microsoft.com/en-us/semantic-kernel/frameworks/agent/agent-orchestration/group-chat.
9. Drammeh, P. (2025). Multi-agent LLM orchestration achieves deterministic, high-quality decision support for incident response. arXiv. https://arxiv.org/abs/2511.15755.
10. Orogat, A., Rostam, A., & Mansour, E. (2026). Understanding multi-agent LLM frameworks: A unified benchmark and experimental analysis. arXiv. https://arxiv.org/abs/2602.03128.
11. Xu, Y., Zeng, B., Qiu, Z., Zhang, Z., Yue, G., Liao, X., Jin, H., & Li, Q. (2025). AgentRace: Benchmarking efficiency in LLM agent frameworks. OpenReview. https://openreview.net/forum?id=eUuxWAQA5F.
12. Kulkarni, S., & Kulkarni, Y. (2026). Benchmarking multi-agent LLM architectures for financial document processing: A comparative study of orchestration patterns, cost-accuracy tradeoffs and production scaling strategies. arXiv. https://arxiv.org/abs/2603.22651.