ECAI 2025
I. Lynce et al. (Eds.)

doi:10.3233/FAIA251440

Multi-Agentic Workflows and Long-Term Memory Use Cases in AI-Builder

Swetha Lakshmana Murthy[a,*,1], **Sangamithra Panneer Selvam**[a,**,1] **and Martin Welß**[a,1]

[a]Fraunhofer IAIS - Institute for Intelligent Analysis and Information Systems, Sankt Augustin, Germany

Abstract. Developments in Artificial Intelligence (AI) today have sparked growing interest in topics such as agentic workflows and Long-Term Memory (LTM) architectures, which extend the capabilities of Large Language Models (LLMs) beyond their current limitations. Agentic workflows represent a significant paradigm shift by enabling LLMs to exhibit goal-oriented behavior, decision-making, and adaptability within dynamic environments. On the other hand, LTM systems enable LLMs to retain and retrieve information across multiple interactions, allowing for personalization, context-aware reasoning, and additional functionalities. This paper elucidates two illustrative use cases, namely the Agentic Event Planner and MemoryGraph, which exemplify the integration of agentic workflows, long-term memory, LLM switching, and related elements into cohesive, hybrid AI pipelines for experimentation purposes. Implemented on the AI-Builder platform, these prototypes benefit from modularity, reusability, and a user-friendly drag-and-drop design environment, with modest orchestration complexity compared to the flexibility and interoperability afforded by the platform.

1 Introduction

Large Language Models (LLMs) are leading transformative advances across domains, yet they remain limited by context window size and the absence of persistent memory [25]. Also, standalone LLMs cannot be used in critical domains due to hallucination and the absence of reproducibility. These constraints hinder their ability to support adaptive, sustainable tasks and personalized interactions. To address these limitations, two complementary research directions have gained traction: agentic workflows and Long-Term Memory (LTM) architectures [6]. Together, they redefine the boundaries of LLM capabilities by enabling systems that can reason, plan, and learn over multiple interactions.

Agentic workflows use formal specifications like pseudocode or structured procedures to guide LLM behavior, balancing flexibility and compliance [22, 26]. In multi-agent and human-in-the-loop settings, role-aware coordination ensures reliable execution and clear responsibilities [4]. The Model Context Protocol (MCP) standardizes interactions between models, tools, and data, enhancing scalability and integration [14, 5]. Complementary microservice architectures improve maintainability and interoperability using standards like gRPC and Protocol Buffers for efficient, cross-language communication [27, 17, 13, 10]. In microservices-based multi-agent systems, agents act as independent services collaborating through protocols like MCP and A2A for seamless coordination [12].

Parallelly, LTM frameworks introduce mechanisms for context preservation, persistent learning, and adaptive behavior. LongMem [24] separates memory storage from computation, enabling scalable recall and personalization, while OMNE [15] treats memory as a foundation to self-improving AI. AIDA-Bot [18] highlights the integration potential of conversational agents with structured knowledge graphs to support knowledge-centric reasoning and retrieval.

Multi-agentic workflows devices a shift from static prompt engineering to structured, goal-oriented behavior that are capable of decision-making, tool use, and coordination. These workflows support complex task execution with modularity and interpretability. In parallel, LTM architectures provide persistent knowledge stores that maintain context across sessions, allowing coherent reasoning and personalization.

In this paper, we present two use cases, **Agentic Event Planner** and **MemoryGraph**, that showcase how agentic and memory-driven pipelines can be composed in practice. These are implemented using AI-Builder [2, 1], which serves as the supporting platform for integrating and orchestrating the underlying components.

2 Platform Architecture Overview: AI-Builder

AI-Builder [2, 1] is an open platform for human-centered hybrid AI workflows, integrating symbolic and neural-network components. As a core element of the European AI-on-Demand Platform [3], it builds on a customized Graphene framework [8, 9] and microservice architecture [21, 20] using RESTful APIs, gRPC, and brokers/streaming protocols, with support for feedback loops in evolutionary AI.

Central to AI-Builder is the Design Studio, a visual environment with a drag-and-connect interface for building AI pipelines with reusable components (nodes) and guided connections, while remaining technology-agnostic. These pipelines are structured as directed cyclic graphs, with edges defining data flow through Protobuf message contracts. Using protoc [11], developers generate gRPC stubs for unary or streaming calls, enabling real-time and batch processing. Pipeline nodes consist of model nodes (referenced by image URIs), data brokers (for real-time input), and shared folders (for data exchange) that communicate through standardized Protobuf messages.

Orchestration Model: The Generic Parallel Orchestrator mediates node communication in a three-party pattern (Node A $\rightarrow$ Orchestrator $\rightarrow$ Node B) where nodes are decoupled and unaware of downstream clients. This supports asynchronous, stateless interactions, improving scalability and component reuse. It can handle both

* Corresponding email: swetha.lakshmana.murthy@iais.fraunhofer.de
** Corresponding email: sangamithra.panneer.selvam@iais.fraunhofer.de
[1] These authors contributed equally to this work.

Nodes in Agentic Event Planner

Event Planner

E1 `rpc SendUserQuery (Empty) returns (stream UserMessage);`
E2 `rpc RecvResponse (stream ToolResult) returns (Empty);`

LLM

L1 `rpc GetUserQuery (stream UserMessage) returns (Empty);`
L2 `rpc GetFuncSchemas (stream FuncSchemasResponse) returns (stream ToolRequest);`
L3 `rpc RecvCallTool (stream ToolInvocationResult) returns (stream ToolInvocationResult);`

MCP-Server-X

A1 `rpc SendFuncSchemas (Empty) returns (stream FuncSchemasResponse);`
A2 `rpc InvokeTool (stream ToolRequest) returns (stream ToolInvocationResult);`

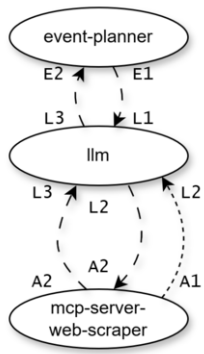

Figure 1. Agentic Event Planner: RPC Interfaces and Connections. Illustrated with a sample MCP Web Scraping Server. Arrows are annotated with interface references; origin/destination symbols indicate the input/output usage of each RPC message.

streaming and non-streaming communication. All microservices run as Kubernetes pods with NGINX ingress and load balancing.

Port Semantics and Protobuf Compatibility: Models expose input ports (white) that consume messages and output ports (black) that emit responses. Connections are permitted only when Protobuf message signatures match in tags, field types, and cardinality (message names can be ignored).

Trade-offs and Benefits of AI-Builder in AI Experimentation: With AI-Builder, the added value for AI researchers and developers is in reducing the design and integration overhead of manually combining heterogeneous AI technologies. Typically, such experiments or workflows require custom code, custom protocols, ad-hoc orchestration, and integration of LLMs, which hinders reuse and adds complexity. By contrast, AI-Builder enforces standardized Protobuf interfaces, enabling type-safe, decoupled interoperability. This simplifies exploration of hybrid AI designs, balancing symbolic and data-driven approaches, and reusing modules across use cases. While Protobuf contracts require some upfront effort, the resulting scalability, modularity, and faster experimentation generally outweigh these costs for rapidly evolving workflows.

3 Agentic and Memory-Driven Pipelines

3.1 Agent-Based Coordination and Execution

As a representative use case, we introduce the *Agentic Event Planner*, a pipeline that enables the composition and orchestration of multiple MCP-based tools for planning travel activities for an event. This pipeline demonstrates coordinated interactions among an Event Planner node, LLM, MCP servers, and tool endpoints, as shown in Figure 2. The LLM functions as a semantic router, interpreting user queries, identifying agent capabilities, and dispatching tool invocations accordingly. This enables distributed yet cognitively coherent decision-making, a central capability supported by AI-Builder. The LLMs can be switched according to user preference, and the MCP tools can be selected as and when required, thus adhering to AI-Builder's flexibility and extensibility for experimenting with diverse pipeline scenarios.

The platform employs a streaming, asynchronous architecture to support real-time orchestration. User queries flow through the `SendUserQuery` Remote Procedure Call (RPC), where the LLM interprets intent, identifies the appropriate `agent_type`, and dynamically selects tools by matching

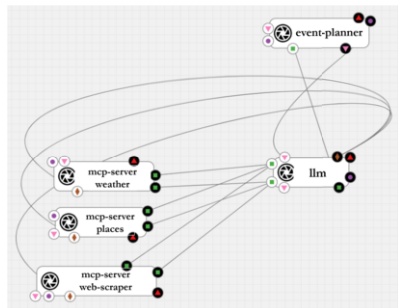

Figure 2. Agentic Event Planner pipeline in AI-Builder showing interactions among the Event Planner, LLM, and MCP servers.

agent-specific function schemas. These schemas are discovered in parallel via the `GetFuncSchemas` and server-streaming `SendFuncSchemas` RPCs. Tools are then invoked over bidirectional `InvokeTool` streams, and structured results are received via `ToolInvocationResult` messages to update the LLM's context. We use OpenAI's function-calling [19] interface for tool selection. Each MCP server runs a non-blocking gRPC server using Python's `grpc.aio` and `asyncio`, enabling high-throughput, concurrent tool execution. Server lifecycles are managed asynchronously to allow graceful shutdown and dynamic service registration. This process is illustrated in the table and in Figure 1.

Message-Driven Orchestration: Communication between the LLM and MCP servers is structured using typed, schema-driven gRPC messages:

- `FuncSchemasResponse` – Advertises an agent's tool functions and JSON schemas, streamed continuously to support dynamic updates (heartbeat mechanism).
- `ToolRequest` – Encapsulates the selected tool name and JSON-formatted arguments as invoked by the LLM.
- `ToolInvocationResult` – Returns structured outputs of tool executions to augment the LLM's contextual reasoning.

The LLM node uses `agent_type` ENUMs to align user intent with agent capabilities, ensuring modular extensibility. Tool schemas are fetched in parallel to reduce orchestration latency, and the results are used to augment the LLM's internal state, enabling refined, grounded responses. This design highlights AI-Builder's support for high-level agent composition, tool discovery, and asynchronous real-time orchestration.

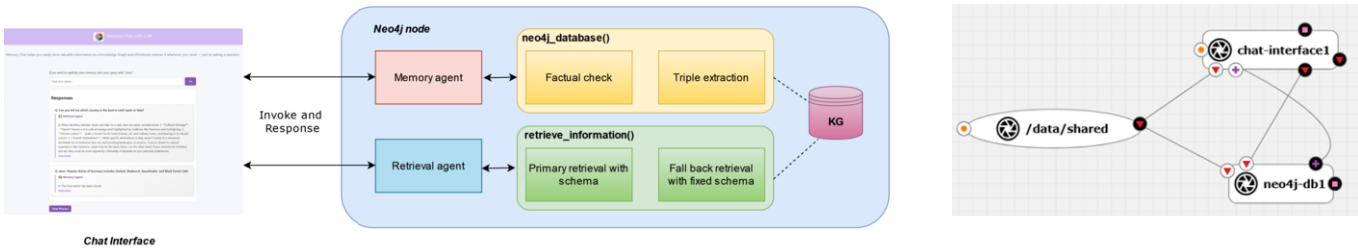

Figure 3. (Left) Architecture for the Long-Term Memory pipeline. (Right) MemoryGraph (LTM) pipeline in AI-Builder.

3.2 Long-Term Memory

The second use case of the paper is the Long-term memory (LTM) for large language models that involves techniques to allow models remember their previous conversations. These techniques help LLMs to utilize information from various interactions beyond their trained knowledge and context window. Exploring diverse approaches, this use case delves into Agentic AI and StateGraph, with MemoryGraph showcasing efficient query-driven storage and retrieval in a knowledge graph.

The use case consist of StateGraph from Langchain framework, Agentic AI, prompt engineering and Knowledge graphs in Neo4j database. A Neo4j-based knowledge graph provides persistent, structured memory, enabling efficient storage and semantic querying [7, 23]. StateGraph, also called as directed acyclic graphs, act as an orchestration layer that models the control flow and decision routes whereas Agent-based LLMs simulate autonomous behavior via prompting techniques [16]. The proposed Stategraph acts as a dynamic workflow that routes user queries through a sequence of decision points and agent invoking, as in Figure 4 . It starts by classifying the query - determining whether it intends to store new information or retrieve existing knowledge using an internal `classify_query_router`. Based on this classification, it either routes the query to invoke the memory agent, which stores data via `neo4j_database` tool, or invoke retrieve agent, which attempts retrieval via a `retrieval_information` tool. If retrieval fails, the system automatically redirects the query to the invoke memory agent path, ensuring continuity in the workflow.

The AI-Builder architecture and pipeline (Figure 3) process user queries by routing them through neo4j-db node, where inputs are classified as either store-type or question/statement.

- For store-type queries, the system invokes the memory agent, which performs a factuality check. If the input is either subjective (*e.g., "LLMs are amazing"*) or factually verifiable (*e.g., "The sun rises in the east"*) or not a question type (*e.g., "Am I allowed to save a project here?"*), it proceeds to a triple and property extraction phase using `neo4j_database()` to store in Neo4j.
- If the input is not a store-type, the system calls the retrieval agent, which attempts to answer the query using a primary retrieval process based on the existing KG schema. If that fails, it uses a fallback retrieval mechanism with a fixed schema via retrieve_information().
- If retrieval still fails, the memory agent is invoked again to store the knowledge for future reference.

Throughout this process, the agents handle execution logic and return the final response to the user via the web interface. The integration of StateGraphs with persistent memory nodes in AI-Builder

Figure 4. Stategraph workflow with Agentic AI Flowchart

demonstrates structured control flow, scalable memory management, and seamless agent execution.

4 Conclusion and Future Work

AI-Builder's agent-based pipeline uses a microservices architecture, where MCP servers expose modular, schema-driven tools coordinated by an LLM-powered semantic router, simplifying integration of external tools. In the memory pipeline, StateGraphs are leveraged as a structured orchestration layer and persistent memory nodes for storing contextual knowledge. The system enables adaptive and context-aware agent behavior with Neo4j as a scalable knowledge graph backend. With AI-Builder, we demonstrate how these components can be seamlessly composed to support intelligent use cases like Agentic Event Planner and MemoryGraph. In conclusion, AI-Builder reduces integration overhead, promotes reuse through standardized Protobuf interfaces, and enables adaptive, context-aware workflows, with schema design effort outweighed by scalability. As part of the AI-on-Demand Platform, it fosters community collaboration. As a future scope, we aim to unify agent workflows and memory systems, supporting contextual personalization and dynamic knowledge retrieval within a single intelligent workflow.

Acknowledgements

This work has been supported by the AI4Europe project that has received funding from the European Union's Horizon Europe research and innovation programme under Grant Agreement nº 101070000.

References

[1] AI4EU Project. AI4EU Experiments Platform. Available at https://aiexp.ai4europe.eu/, 2022. Accessed: 2025-08-25.

[2] AI4EU Project. AI4EU AI-Builder Platform. Available at https://www.ai4europe.eu/ai-builder, 2024. Accessed: 2025-08-25.

[3] AIoD. Ai-on-demand platform. Available at https://www.ai4europe.eu, 2025. Accessed: 2025-08-25.

[4] A. Ait, J. L. C. Izquierdo, and J. Cabot. Towards modeling human-agentic collaborative workflows: A bpmn extension, 2024. URL https://arxiv.org/abs/2412.05958.

[5] Anthropic. Model Context Protocol (MCP). https://modelcontextprotocol.io/introduction, 2024. Accessed: 2025-06-01.

[6] S. Barua. Exploring autonomous agents through the lens of large language models: A review. 2024. doi: 10.48550/ARXIV.2404.04442. URL https://arxiv.org/abs/2404.04442.

[7] Y. Chen, H. Li, H. Li, W. Liu, Y. Wu, Q. Huang, and S. Wan. An overview of knowledge graph reasoning: Key technologies and applications. *Journal of Sensor and Actuator Networks*, 11(4):78, 2022. doi: 10.3390/jsan11040078. URL https://www.mdpi.com/2224-2708/11/4/78.

[8] Eclipse Foundation. Eclipse Graphene. Available at https://projects.eclipse.org/projects/technology.graphene, 2021. Accessed: 2025-08-25.

[9] Eclipse Foundation. Eclipse Graphene. Available at https://gitlab.eclipse.org/eclipse/graphene, 2021. Accessed: 2025-08-25.

[10] Google. Protocol Buffers. Available at https://developers.google.com/protocol-buffers, 2008. Accessed: 2025-08-25.

[11] Google. Protocol buffers compiler (protoc). Available at https://github.com/protocolbuffers/protobuf, 2025. Accessed: 2025-08-25.

[12] M. Goyal and P. Bhasin. Moving from monolithic to microservices architecture for multi-agent systems. *World Journal of Advanced Engineering Technology and Sciences*, 15(1):2119–2124, Apr. 2025. ISSN 2582-8266. doi: 10.30574/wjaets.2025.15.1.0480. URL http://dx.doi.org/10.30574/wjaets.2025.15.1.0480.

[13] gRPC Authors. gRPC: A high performance, open-source universal RPC framework. Available at https://grpc.io/, 2016. Accessed: 2025-08-25.

[14] X. Hou, Y. Zhao, S. Wang, and H. Wang. Model context protocol (mcp): Landscape, security threats, and future research directions, 2025. URL https://arxiv.org/abs/2503.23278.

[15] X. Jiang, F. Li, H. Zhao, J. Wang, J. Shao, S. Xu, S. Zhang, W. Chen, X. Tang, Y. Chen, M. Wu, W. Ma, M. Wang, and T. Chen. Long term memory: The foundation of ai self-evolution, 10 2024.

[16] LangChain. Long-term memory agent documentation. https://python.langchain.com/docs/versions/migrating_memory/long_term_memory_agent/, 2024. Accessed: 2025-05-31.

[17] F. Matos, P. Rego, and F. Trinta. An empirical study about the adoption of multi-language technique in computation offloading in a mobile cloud computing scenario. pages 207–214, 01 2021. doi: 10.5220/0010437802070214.

[18] A. Meloni, S. Angioni, A. Salatino, F. Osborne, D. Reforgiato Recupero, and E. Motta. Integrating conversational agents and knowledge graphs within the scholarly domain. *IEEE Access*, 11:22468–22489, 2023. doi: 10.1109/ACCESS.2023.3253388.

[19] OpenAI. OpenAI Function Calling API. Available at https://platform.openai.com/docs/guides/gpt/function-calling, 2024. Accessed: 2025-08-25.

[20] G. Rehm, D. Galanis, P. Labropoulou, S. Piperidis, M. Welß, R. Usbeck, J. Köhler, M. Deligiannis, K. Gkirtzou, J. Fischer, C. Chiarcos, N. Feldhus, J. Moreno-Schneider, F. Kintzel, E. Montiel, V. Rodríguez Doncel, J. P. McCrae, D. Laqua, I. P. Theile, C. Dittmar, K. Bontcheva, I. Roberts, A. Vasiljevs, and A. Lagzdiņš. Towards an interoperable ecosystem of AI and LT platforms: A roadmap for the implementation of different levels of interoperability. In G. Rehm, K. Bontcheva, K. Choukri, J. Hajič, S. Piperidis, and A. Vasiljevs, editors, *Proceedings of the 1st International Workshop on Language Technology Platforms*, pages 96–107, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-64-1. URL https://aclanthology.org/2020.iwltp-1.15/.

[21] P. Schüller, J. P. Costeira, J. Crowley, J. Grosinger, F. Ingrand, U. Köckemann, A. Saffiotti, and M. Welss. Composing complex and hybrid ai solutions, 2022. URL https://arxiv.org/abs/2202.12566.

[22] Y. Shi, S. Cai, Z. Xu, Y. Qin, G. Li, H. Shao, J. Chen, D. Yang, K. Li, and X. Sun. Flowagent: Achieving compliance and flexibility for workflow agents, 2025. URL https://arxiv.org/abs/2502.14345.

[23] L. Tian, X. Zhou, Y.-P. Wu, W.-T. Zhou, J.-H. Zhang, and T.-S. Zhang. Knowledge graph and knowledge reasoning: A systematic review. *Journal of Electronic Science and Technology*, 2023. URL <insertURLifavailable>. Invited Paper.

[24] W. Wang, L. Dong, H. Cheng, X. Liu, X. Yan, J. Gao, and F. Wei. Augmenting language models with long-term memory, 2023. URL https://arxiv.org/abs/2306.07174.

[25] X. Wang, M. Salmani, P. Omidi, X. Ren, M. Rezagholizadeh, and A. Eshaghi. Beyond the limits: A survey of techniques to extend the context length in large language models, 2024. URL https://arxiv.org/abs/2402.02244.

[26] R. Xiao, W. Ma, K. Wang, Y. Wu, J. Zhao, H. Wang, F. Huang, and Y. Li. Flowbench: Revisiting and benchmarking workflow-guided planning for llm-based agents, 2024. URL https://arxiv.org/abs/2406.14884.

[27] L. Zhang, K. Pang, J. Xu, and B. Niu. High performance microservice communication technology based on modified remote procedure call. *Scientific Reports*, 13, 07 2023. doi: 10.1038/s41598-023-39355-4.