Agentic Hyperautomation: A Distributed Architecture for Scalable AI-Driven Workflows

Arnaldo Tomasino[1], Saverio Ieva[1,2], Giuseppe Loseto[2,3], Floriano Scioscia[1,2], *Senior Member, IEEE*, Michele Ruta[1,2], *Member, IEEE*, Angelo Ingianni[1,4], Marco Minoia[4], and Gianmarco Genchi[4]

***Abstract*— Hyperautomation aims to digitize end-to-end business processes, but most of the current solutions and platforms still rely on pre-defined workflows to carry out complex tasks. However, business process automation can greatly benefit from recent technological innovations at the intersection of Large Language Models (LLMs) and Multi-Agent systems (MAS). In this paper, we present an Agentic Artificial Intelligence (AI) framework where a central LLM-driven orchestrator dynamically plans and delegates tasks to specialized LLM agents, which in turn exploit tools to act on business platforms and other external systems. A case study based on document management illustrates how the approach is able to deal with complex requests involving source file parsing and report generation, leveraging an approach based on Retrieval Augmented Generation (RAG) to enable knowledge sharing among specialized agents. Finally, the proposed framework introduces a practical blueprint for scalable, explainable Agentic AI in enterprise hyperautomation environments.**

I. Introduction

The term Hyperautomation encompasses different technological fields such as Business Process Management (BPM), Robotic Process Automation (RPA), Process Mining, Artificial Intelligence (AI), Low-Code Development, and Integration Platform as a Service (iPaaS). Hyperautomation goals focus on productivity, cost savings, error reduction, process quality improvement and customers satisfaction [1]. To automate actions and decisions, Hyperautomation technologies have increasingly integrated AI algorithms [2] [3], leading to Intelligent Business Process Management (iBPM) or Intelligent Robotic Process Automation (iRPA) / Intelligent Process Automation for RPA [4] [5].

Large Language Models (LLMs) and intelligent agents will be ubiquitous in Hyperautomation tools, automating complex workflows and cognitive tasks [6] [7]. Agentic AI enables autonomous AI systems to make decisions and act toward complex goals with limited or no supervision [8]. It merges LLM flexibility with programming accuracy, for reasoning and planning, using external tools, memory mechanisms and advanced knowledge representation and retrieval, for independent goal achievement [9].

This paper introduces a framework based on LLM and Retrieval Augmented Generation (RAG) for coordination of intelligent agents in the context of Hyperautomation. The proposed architecture enables dynamic handling of complex requests, without predefined workflows. An LLM generates atomic tasks and delegates them to specialized agents, which interact with business systems via dedicated tools for flexible integration. While the approach is conceptual, practical deployment requires validating LLM behavior, testing RAG reliability, and assessing scalability in realistic settings.

The remainder of the paper is organized as follows. Section II provides an overview of related works in the reference domain and Section III describes in detail the proposed framework. A case study based on the proposed architecture is presented in Section IV. Finally, Section V outlines conclusions and future works.

II. Related work

Different Agentic AI development methodologies are analyzed in [9], ranging from architecture, learning paradigms, training and evaluation techniques. The survey emphasizes that the Multi-Agent System (MAS) model presents challenges regarding scalability, and federated learning and hierarchical structures are suitable for improving on that front. Moreover, transparency and explainability are crucial to understand and govern agents. The functional components of the Agentic AI paradigm, *i.e.*, perceptions, planning, memory, and actions, become all LLM-based, as described in [10] [11], highlighting the critical need to evaluate the safety and task planning capabilities due to the possibility of hallucinations.

The AI-Augmented BPMS (ABPMS) manifesto [12] describes functional and architectural requirements in the full lifecycle of emerging Business Process Management System (BPMS), that are context-sensitive thanks to AI. Execution flow is not pre-determined, therefore proactive behavior, discovering opportunity and validation are performed at run-time. The purpose of the manifesto is to outline a proposal and provide recommendations, but without details of architectural design patterns for implementation, which is the purpose of our work. As a research gap, the manifesto outlines the need to support (i) processes with incomplete structure at design time that emerge at run-time, (ii) the ability to incorporate new knowledge and tasks at run-time and (iii) explainable output or recommendation to the user.

*The research was supported by *Digital Enterprise* grant (code NIL6S28), co-funded by Lutech S.p.A. and European Regional Development Fund for Apulia Region 2014/2020 Operating Program.

[1] Department of Electrical and Information Engineering, Polytechnic University of Bari, via E. Orabona 4, I-70125 Bari, Italy; {name.surname}@poliba.it

[2] donkeyPower S.r.l., via E. Orabona 4, I-70125 Bari, Italy; {name.surname}@donkeypower.it

[3] Department of Engineering, LUM "Giuseppe Degennaro" University, S.S. 100 km 18, I-70010 Casamassima, Italy; loseto@lum.it;

[4] Lutech S.p.A., via M. Gorki 30/32C, I-20092 Cinisello Balsamo, Italy; {name.surname}@lutech.it

In [7], starting from a textual description of process model and some solved examples, it is possible to derive both the imperative (*e.g.*, Linear Temporal Logic notation) and declarative description (e.g. BPM notation) of a business process by using LLMs. A key limitation is the manual prompt tuning required to address common issues, highlighting the value of the agent-oriented approach adopted in this work.

Large Process Models [6], integrating LLMs with symbolic information (*e.g.*, knowledge graphs) and reasoning, are a valid approach to handle unpredictable LLMs output, as outlined in [12] and [13]. LLMs in Process Mining [14] enable domain knowledge as well as logical and temporal reasoning for anomaly detection, analysis and process enhancement. Moreover, an agentic approach supports hypothesis generation from event logs and text-to-query translation for their verification.

From the point of view of workflow automation, LLMs and agents are used to generate executable workflow with dynamic decision-making during execution. FlowMind [15] adopts a two-phase approach. In the first phase, the LLM is instructed with domain information, expected task, and details about available Application Programming Interfaces (APIs). In the second human-in-the-loop phase, the LLM is prompted to generate Python code workflows using the provided APIs. Although this phase could benefit from a MAS, relying on private enterprise APIs helps avoid exposing sensitive data and business rules. LLM4Workflow [16] adopts a similar strategy, but it is improved with RAG and automatic workflow evaluation. One of the most advanced approaches is ProAgent [17]: similarly to our proposal, it uses specialized LLM agents (DataAgent and ControlAgent) to generate a workflow, whose output is Python code that handles the orchestration of tasks. interfacing with the *n8n* (`https://n8n.io/`) platform to complete the required business tasks.

Compared with the above works, the proposed approach introduces a unified architecture that combines: (i) scalable event-driven multi-agent orchestration; (ii) memory-augmented RAG for context-aware task planning; (iii) seamless integration with heterogeneous external business platforms via plugin-based tools.

III. Multi-Agent Architectural Design for Hyperautomation

The system consists of the following layers (numbered in Figure 1):

1) **orchestration layer**: an *Orchestrator Agent* receives a request and uses an LLM to generate a workflow and distribute tasks among agents;
2) **agent definition layer**: each agent is highly specialized in a particular task and has a distinct role in the overall infrastructure;
3) **tools layer**: tools define additional functionality a given agent must use to interact with external systems;
4) **plugins layer**: tools can be implemented as general interfaces provided to agents, relying on plugins to adapt their actions to specific external systems.

A. Agents' Role and Interactions

In the outlined architecture, each agent (depicted in layer 2 of Figure 1) leverages an LLM to carry out a specific task. An agent is defined by:

- a detailed system prompt, allowing the agent to instruct the LLM on how to perform a particular processing step in the intended way, *i.e.*, defining the expected inputs, outputs and task execution requirements;
- a task description, used to define the agent's role w.r.t. external components.

The **Orchestrator Agent** acts as the main interface to external actors using the system. Upon receiving a request describing a complex task, it invokes the LLM to design a specific action plan by subdividing the complex task into minimal steps. Then, it distributes tasks to agents relying on an *Agents Catalog*, which contains names and descriptions of each specialized agent in the system. This allows the orchestrator to dynamically coordinate the execution by invoking a specific set of agents until the completion of the requested workflow. At each step:

- the **input** to the (set of) specialized agent(s) is a prompt automatically generated by the Orchestrator Agent;
- the **output** from the agent(s) includes messages that contain the produced output, as well as the status of the execution, which can be successful or indicate an exception. Based on the output of each invoked agent, the orchestrator selects the next one. In case of an exception, the Orchestrator Agent is capable of redirecting the flow of interactions to the appropriate agents to solve it.

Moreover, the system agents can be equipped with *memory*. The **Long-Term Memory** stores relatively stable information that changes infrequently. It consists of a *Semantic Memory*, holding business information, and an *Episodic Memory* storing details about past successful workflows. When receiving a request, the Orchestrator Agent may select one of the existing sequences of interaction with specialized agents in the *Episodic Memory* which best matches the requested task, instead of creating a completely new workflow.

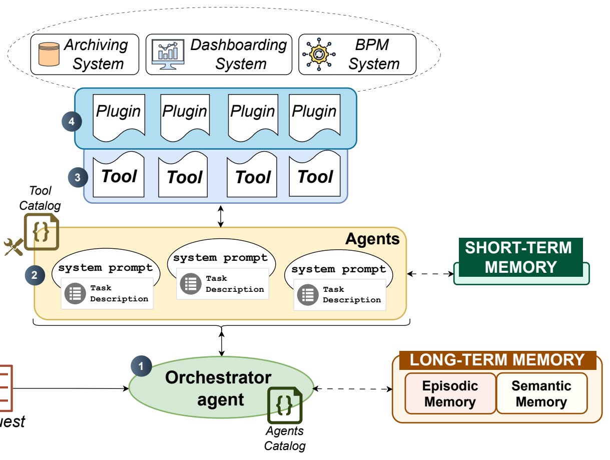

Fig. 1. Architecture of the proposed framework

This approach can significantly reduce computational load, as the Orchestrator Agent can adapt to manage a given set of requests and become more efficient over time.

Conversely, **Short-term Memory** is a temporary repository that stores domain-specific data related to the current transaction. This information serves as shared contextual knowledge, enabling agents to better tailor the LLMs' behavior during task execution.

B. Integration with External Systems

Tools connect agents to external systems both to obtain data and information as input to their execution and to produce appropriate output. For instance, output tools can interface with dashboarding systems to visualize business metrics from multiple Enterprise Resource Management (ERP) and BPM platforms or generate detailed and comprehensive reports in various formats. Moreover, tools can interact with BPM systems to invoke predefined business processes or methods as needed. This allows agents to retrieve information from connected platforms, execute existing business rules for data validation, or autonomously update their state based on their own decision, establishing a feedback loop that ensures interoperability across platforms.

Tools are organized in a *Tools Catalog*, containing their details, *i.e.*, their names, needed inputs and expected outputs, permissions and activation details, such as HTTP endpoints. Moreover, in the proposed system, tools do not directly handle platform-specific operations but they are designed as interfaces defining a set of generic operations to interact with a given category of external systems, abstracting away from platform-specific details. Instead, they rely on *plugins* to connect to particular instances of integrated BPM/ERP external platforms. *Plugins* are the only components that use specific libraries to adapt generic operations to a given external system.

Furthermore, the proposed system can be integrated into existing business automation platforms as an **event-driven component**. Existing business platforms must implement mechanisms to send messages to the Orchestrator Agent whenever events occur, with produced data adapted for inclusion in the prompt passed to said agent. Then, the LLM-orchestrated execution of tasks can be carried out in an asynchronous way. Thus, the proposed system can act as an additional service within existing hyperautomation environments. responding to events by orchestrating asynchronous distributed services and interacting with external software to produce the appropriate output.

IV. Workflow Automation via Agentic Coordination

The proposed architecture can be adapted for workflow automation using an agent-based environment in several business scenarios that are being explored by *Lutech S.p.A.* with their clients. The document management case is shown in Figure 2, highlighting additional components.

The system operates on an event-driven basis, where each agent is both an event producer and a consumer. All

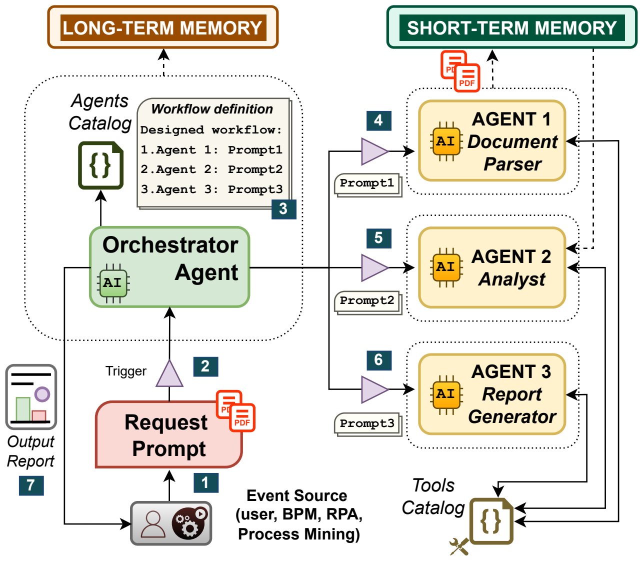

Fig. 2. Workflow automation

internal and external requests trigger events that activate specific components, ensuring a consistent communication pattern. Each event is routed through a different channel, each managed according to specific rules.

This design allows the system to dynamically generate workflows in a flexible and non-deterministic manner, enabling fully automated resolution of user requests.

A. Use Case: LLM-based Document Management

This section describes a reference use case involving the generation of a business report based on a user transaction. To support this process, the system leverages two types of memory. The semantic memory stores stable business information like product backlogs, guidelines, required formats, and policies. The episodic memory holds details of past successful workflows, potentially including user prompts, workflow history, and labels for easy topic retrieval.

Considering the numbered steps in the workflow in Figure 2, the process begins with a request (1) originating from an external BPM system or a user interface. This request, accompanied by relevant input documents, serves as the triggering event for the execution of the automated system, which is orchestrated and supervised by the Orchestrator Agent (2). The orchestrator first queries the episodic memory to determine whether a similar task has been successfully completed in the past. If no suitable precedent is found, it generates a new workflow from scratch. To this end, it performs agentic RAG on the semantic memory, using the user prompt to retrieve the relevant business knowledge required for planning. The result is a structured workflow definition that specifies which agent should execute each task (3). The execution of the first agent in the plan, the Document Parser (4), is then triggered. This agent analyzes the input documents and stores the extracted information in a short-term working memory, using the prompt provided by the orchestrator along with appropriate tools. Upon completion, an event notifies the orchestrator, which reviews the output. If the result is satisfactory, the orchestrator activates the next agent in the sequence, the Analyst (5). This agent performs agentic RAG on the parsed content stored in short-term memory. Based on the prompt provided by the orchestrator, it extracts information relevant for drafting the report. The orchestrator, after validating these results, proceeds to trigger the Report Generator agent (6), providing a prompt that specifies the report requirements, based on both the user request and retrieved business knowledge. This agent employs tools and its LLM to generate and execute the code needed to produce the report, which is subsequently returned to the orchestrator. After final validation, the orchestrator sends the report to the external system that made the initial request.

B. Scalability and Performance Considerations

The proposed agent-based framework inherently suggests a scalable, resilient, and flexible infrastructure capable of managing unpredictability in terms of both workflow requirements and workload volume. Agents work independently and are highly decoupled, but also need to access a shared context for reasoning and making decisions. An event-driven paradigm facilitates asynchronous and independent access, modification, and storage of context information. Consequently, agents communicate via channels, eliminating direct constraints and dependencies, thereby enhancing flexibility and enabling scalability. The Event-Driven Architecture (EDA) paradigm aligns well with cost efficiency, as system components are triggered only by events, making resource usage proportional to actual demand. This naturally supports deployment as scalable, serverless functions activated on demand. The resulting architecture facilitates decoupling, cost reduction, and operational flexibility, which are key enablers for modern agent-based business environments. However, real-world deployments face challenges such as inconsistent behaviors across LLM agents, efficient cluster orchestration, and dynamic resource allocation. Robust communication, standardized tool invocation – e.g., via Model Context Protocol (MCP) – and rollback mechanisms are key for reliable and recoverable workflows.

V. Conclusions

This paper has introduced an LLM-based MAS for hyperautomation. An Orchestrator Agent uses an LLM to generate a structured workflow composed of a set of atomic tasks, starting from a complex request involving business-related processes, allowing the system to autonomously adapt to various tasks, without the need of human intervention and with the additional capability to reuse already prepared workflows. Then, tasks are distributed to specialized agents, which also make use of an LLM. This enhances overall scalability and makes it possible for the system to be easily integrated into existing hyperautomation platforms as an event-driven component. In order to demonstrate the advantages of the integration of the system in a business automation environment, a case study focused on document management and report generation has been described. In future work, human-in-the-loop patterns will be considered for validation, refinement or correction of complex workflows.

References

[1] N. Neis, E. Kathol, and A. Winkelmann, "Structuring intelligent automation: A hyperautomation taxonomy," in *Proceedings of the 58th Hawaii International Conference on System Sciences*, 2025, pp. 5663–5672.

[2] S. Weinzierl, S. Zilker, S. Dunzer, and M. Matzner, "Machine learning in business process management: A systematic literature review," *Expert Systems with Applications*, p. 124181, 2024.

[3] A. Mathew and H. Alex, "Hyper automation and augmented intelligence," in *2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT)*. IEEE, 2023, pp. 1230–1234.

[4] S. Mohanty, S. Vyas, S. Mohanty, and S. Vyas, "Intelligent Process Automation = RPA + AI," *How to Compete in the Age of Artificial Intelligence: Implementing a Collaborative Human-Machine Strategy for Your Business*, pp. 125–141, 2018.

[5] S. Waduge, R. Sugathadasa, A. Piyatilake, and S. Nanayakkara, "A process analysis framework to adopt intelligent robotic process automation (IRPA) in supply chains," *Sustainability*, vol. 16, no. 22, p. 9753, 2024.

[6] T. Kampik, C. Warmuth, A. Rebmann, R. Agam, L. N. Egger, A. Gerber, J. Hoffart, J. Kolk, P. Herzig, G. Decker *et al.*, "Large Process Models: A Vision for Business Process Management in the Age of Generative AI," *KI-Künstliche Intelligenz*, pp. 1–15, 2024.

[7] M. Grohs, L. Abb, N. Elsayed, and J.-R. Rehse, "Large language models can accomplish business process management tasks," in *International Conference on Business Process Management*. Springer, 2023, pp. 453–465.

[8] L. Hughes *et al.*, "Ai agents and agentic systems: A multi-expert analysis," *Journal of Computer Information Systems*, pp. 1–29, 2025.

[9] D. B. Acharya, K. Kuppan, and B. Divya, "Agentic ai: Autonomous intelligence for complex goals–a comprehensive survey," *IEEE Access*, 2025.

[10] J. He, R. Ghosh, K. Walia, J. Chen, T. Dhadiwal, A. Hazel, and C. Inguva, "Frontiers of large language model-based agentic systems-construction, efficacy and safety," in *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, 2024, pp. 5526–5529.

[11] C. Chawla, S. Chatterjee, S. S. Gadadinni, P. Verma, and S. Banerjee, "Agentic AI: The building blocks of sophisticated AI business applications," *Journal of AI, Robotics & Workplace Automation*, vol. 3, no. 3, pp. 1–15, 2024.

[12] M. Dumas, F. Fournier, L. Limonad, A. Marrella, M. Montali, J.-R. Rehse, R. Accorsi, D. Calvanese, G. De Giacomo, D. Fahland *et al.*, "AI-augmented business process management systems: a research manifesto," *ACM Transactions on Management Information Systems*, vol. 14, no. 1, pp. 1–19, 2023.

[13] S. Ieva, D. Loconte, G. Loseto, M. Ruta, F. Scioscia, D. Marche, and M. Notarnicola, "A retrieval-augmented generation approach for data-driven energy infrastructure digital twins," *Smart Cities*, vol. 7, no. 6, pp. 3095–3120, 2024.

[14] A. Berti, H. Kourani, H. Häfke, C.-Y. Li, and D. Schuster, "Evaluating large language models in process mining: Capabilities, benchmarks, and evaluation strategies," in *International Conference on Business Process Modeling, Development and Support*. Springer, 2024, pp. 13–21.

[15] Z. Zeng, W. Watson, N. Cho, S. Rahimi, S. Reynolds, T. Balch, and M. Veloso, "FlowMind: automatic workflow generation with LLMs," in *Proceedings of the Fourth ACM International Conference on AI in Finance*, 2023, pp. 73–81.

[16] J. Xu, W. Du, X. Liu, and X. Li, "LLM4Workflow: An LLM-based Automated Workflow Model Generation Tool," in *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, 2024, pp. 2394–2398.

[17] Y. Ye, X. Cong, S. Tian, J. Cao, H. Wang, Y. Qin, Y. Lu, H. Yu, H. Wang, Y. Lin *et al.*, "ProAgent: From Robotic Process Automation to Agentic Process Automation," *arXiv preprint arXiv:2311.10751*, 2023.