Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 142 (2026) 974–979

www.elsevier.com/locate/procedia

36th CIRP Design Conference (CIRP Design 2026)

Designing Multi-LLM-Agent Systems for Material Flow Simulation in Production Systems

Merlin Korth[a]*, Martin Benfer[a], Gisela Lanza[a]

[a]*wbk Institute of Production Science, Karlsruhe Insitute of Technology (KIT), Kaiserstraße 12, 76131 Karlsruhe, Germany*

* Corresponding author. Tel.: +49 1523 9502565. *E-mail address:* merlin.korth@kit.edu

Abstract

The increasing complexity of market environments, characterized by reduced batch sizes and a growing diversity of product variants due to mass personalization, demands greater adaptability and flexibility from production systems. To effectively manage this complexity, material flow simulation can be utilized. However, since simulation software is an expert tool, the entry barrier for inexperienced production planners remains high. This barrier can be lowered by developing virtual assistants that minimize manual effort and support informed decision-making in dynamic production contexts. Recent advances in large language models offer considerable potential for automating tasks, particularly in production system planning and control. However, due to the complexity of the underlaying task, large language model-based single-agent systems often struggle to solve complex, multi-step problems reliably. To address this limitation, multi-agent systems have been proven to be appropriate. Therefore, new system architectures are required for large language model-based multi-agent systems. Despite the relevance of this topic, there is currently a limited amount of research on the design and implementation of such architectures within the context of simulation studies.
This approach introduces a multi-agent system architecture that enables the supported execution of material flow simulation studies. The behavior of the multi-agent system is analyzed to understand the potential and challenges of how such systems can serve as virtual assistants for production planners conducting simulation studies. A key application is the integration of these large language model-based agents into digital twins of production systems, thereby enabling time savings through supported model adaptation, lowering entry barriers for inexperienced users, enhancing decision support, and reducing error susceptibility through structured procedures. The findings offer insights into architectural design, tool use, and memory mechanisms that support simulation studies and mitigate hallucinations in large language models.

Keywords: Large Language Model, Multi-Agent System, Material Flow Simulation, Software Architecture,

1. Introduction

In the context of Industry 4.0, the structural and operational complexity of cyber-physical production systems has increased considerably. Simulation is a central capability for exploring different layout options, maintaining throughput, and assessing the impact of control policies before they are deployed [1]. However, despite the advancements in simulation technology, expert-centric tools continue to dominate state-of-the-art approaches. The implementation of suitable simulation studies is often time-consuming, and there are only a limited number of specialists available. Furthermore, less experienced teams encounter difficulties with conceptual modelling choices and uncertainty regarding the exploration of problem and solution spaces [2-4]. These limitations hinder the broader, routine utilization of simulation in day-to-day decision-making processes, excluding the domain of simulation experts [2-4]. To facilitate greater accessibility, there is a need for virtual assistants that can translate the challenges and goals of production planning into simulation experiments and guide non-expert users through model setup, scenario definition, and result interpretation, thereby enabling reliable simulation runs [5]. Recent results indicate that language-model-based pipelines can transform natural-language problem statements into executable simulation models for logistics processes, suggesting a more accessible path for virtual assistants [5].

10.1016/j.procir.2026.05.378

Here, agentic large language models (LLMs) can invoke external functions and APIs to integrate the outputs of, e.g., human-made software tools into answers, enabling tool-augmented and comprehensible responses [6].

In this paper we introduce a multi-LLM-agent architecture framework to conduct material flow simulation studies. The developed concept is evaluated using a question catalog with varying complexity of the questions to assess key challenges of the design and thus the applicability of this architecture.

The remainder of this paper is divided into four sections: Section 2 introduces the related work, and Section 3 presents a multi-LLM-agent architecture framework. Section 4 discusses a use case and the proposed architecture. Section 5 concludes with a summary of the main findings and contributions, as well as future research directions.

2. Related Work

In Section 2.1, the concept of digital twins of production systems is introduced, and in Section 2.2, approaches utilizing LLMs in production planning are discussed. In Section 2.3, approaches utilizing agentic LLMs, i.e., LLMs capable of using tools such as simulation models, are explored.

2.1. Digital Twins of Production Systems

As production systems constantly evolve, production planners are required to regularly make decisions about adjusting the production system accordingly [7]. Here, discrete-event simulation, e.g. in form of material flow simulation, offers a suitable approach for analyzing and evaluating these scenarios. The procedure of simulation studies is described by e.g. VDI 3633 [8]. As shown in Figure 1, VDI 3633 explains the steps that material flow simulation studies follow in detail. During the task definition phase, the objectives of the simulation are specified, and the scope of the task is clarified. The subsequent step is a system analysis, in which the elements of the production systems are described and the simulation model is checked to see if it can answer the questions. Next, a model formalisation phase ensues, wherein the descriptions of the simulation model and the simulation study are formalised and subsequently implemented. The collection and preparation of data provides the required input for the study, which involves the acquisition of raw data and their processing into a form suitable for simulation. It is worth noting that these data-related activities must be completed before implementing the simulation model. [8]

When material flow simulation models are deployed as a digital twin, the simulation model can preserve high accuracy over extended periods, providing reliable long-term insights into the production system. [9, 10] Enhancing digital twins to respond quickly and adapt to changing conditions, they enable the projection of system behavior over short- and medium-term horizons, thereby supporting the development of effective production control strategies. [11] The work of Villalonga et al. [12] introduces a framework that leverages the aggregation of multiple digital twins of distinct physical assets with autonomous decision-making and a coordinating digital twin, to enable on-demand optimization of production scheduling.

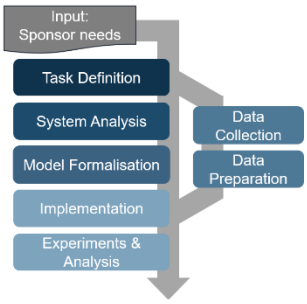

Figure 1: The procedure of simulation studies based on VDI 3633 [22]

2.2. Large Language Models in Production Planning

In manufacturing environments, LLMs have been utilized as tools for managing and organizing domain-specific knowledge [13, 14]. Often, the applications of these technologies have been focused on addressing queries related to maintenance or quality concerns. In the context of production control, LLMs have been employed to forecast equipment failures and unplanned downtimes by utilizing historical records and annotations from personnel [15]. Despite their use in detecting anomalies at the system level, the integration of LLMs into production planning processes and simulation studies remains limited [16].

2.3. Agentic Large Language Models in Production Systems

When introducing agentic systems, the term 'agent' is defined as "anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors" [17]. A multi-agent system is defined as a system in which multiple individual agents interact cooperatively or competitively to achieve a global objective [18]. This is similarly true for agentic LLM systems.

Recent research highlights a range of architectural strategies for integrating large language models into industrial systems. Multi-LLM architectures are exemplified by Zhao et al. [19], who implemented layered LLM components for engine control, and negotiation of physical resources in flexible manufacturing, achieving notable reductions in makespan for multi-variety, small-batch production. Modular multi-agent approaches, where product and resource agents coordinated by a central LLM successfully planned for a two-step CNC processes and high performance in similar complex tasks [20].

Digital twin integration was evaluated by Xia et al. [21], reporting 96% executable skill sequences in automotive assembly and process industry scenarios. Hybrid solutions have demonstrated effectiveness in domain-specific contexts, as shown in warehouse management using a hybrid-2 architecture that combines centralized planning with local LLM agents [22]. Cloud-based designs enable a centralized LLM planner to coordinate device-specific agents, improving task accuracy in logistics and inspection tasks while reducing communication overhead [23].

Unlike the engine-control, and negotiation of physical resources focused architecture of [19], the CNC Process control of [20] or the warehouse management agents [22], our architecture is workflow-driven and designed to provide full simulation lifecycle support within a digital twin.

3. The proposed Multi-LLM-agent Architecture

Developing a multi-LLM-agent architecture requires several key concepts that define the inner processing of information and the interaction between multiple agents, as illustrated in Figure 2. Building on the definition of multi-agent systems of Section 2.3, it is necessary to define the inner logic of an agent and its tools to manipulate the environment (Section 3.1), the environment in which the agent acts and receives information (Section 3.2), and the coordination of multiple agents (Section 3.3).

3.1. Agent Design

Several aspects need to be considered when modeling the inner logic of an LLM agent. Selecting an LLM model through model providers such as Google, OpenAI or Llama defines the agent's core capabilities and determines its existing knowledge. The quality of an LLM in solving specific tasks can be improved through instruction-based or unsupervised fine-tuning. With greater model complexity and increased training effort, it typically leads to more accurate, robust, and contextually appropriate outputs.

3.1.1. The inner 'Thinking' Mechanism

The proposed agent architecture necessitates retrieving further information on which it was not trained. This is achieved with *Retrieval-Augmented Generation* (RAG). RAG utilizes a vector database in which information can be stored and retrieved based on search queries. Therefore, this system allows descriptions of all related manufacturing databases, the simulation model, the production system and any other relevant information to be integrated. The RAG system can also provide exemplary conversations for users with different levels of experience. It can thereby provide blueprints for solving several user queries. [25]

The actual inner logic that is applied to address user queries in our agents is the *Reason and Act* (ReAct) approach, which has been shown to significantly improve accuracy at tasks such as question answering. ReAct integrates explicit information retrieval and evaluation steps with potential tool use to iteratively plan, act, and observe, thereby enhancing accuracy and decision-making capabilities [26]. This provides LLM agents with the knowledge of how to access and use to tools they need to acquire additional information when necessary.

Furthermore, prompt engineering represents a critical design consideration that must be addressed in any LLM and, therefore, any LLM-based agent [27]. The central issue that needs to be addressed here, is that each agent must execute their designated task in a self-contained manner, without unnecessary duplication or deviation. We, therefore, designed a range of prompts for each agent, incorporating examples, recommended tool choices, suggested formulations for answers, and anticipated outcomes for queries.

3.1.2. Memory Mechanism

To integrate information from previous steps and user queries, an LLM needs to integrate a storage module to collect previous communications, past tool calls and their results. However, our preliminary studies have shown that providing the full state ensures complete context but risks to confuse the active agent with irrelevant messages and token costs increase as the amount of context to be processed grows. Passing only the initial or latest query avoids these issues, but causes agents to attempt tasks outside their scope, as they lack awareness of previous progress. An intermediate approach is to forward only the most recent message, requiring each agent to generate a summary of completed and remaining tasks for the successor.

As indicated by our preliminary results, a reduced state space has been shown to be beneficial. Therefore, we created a central state description in which we collect information about the production system, the simulation model, the results of tool calls, the primary query investigated and past conversations. This central state model will be updated after each interaction with the user or a tool call. From the central state description, the necessary context information is extracted and compiled for each agent individually, depending on the agent's tasks.

3.1.3. Invoking Tools

To solve the agent's task, each agent has access to one or several tools that can be invoked via API calls. Here, the capacity of agents to invoke tools for interaction with their environment, referred to as the "simulation model", is determined by the specific tasks and steps allocated to them, as shown in Figure 3. In our case, the System Analyst Agent and the Data Agent have access to the same tools. However, the

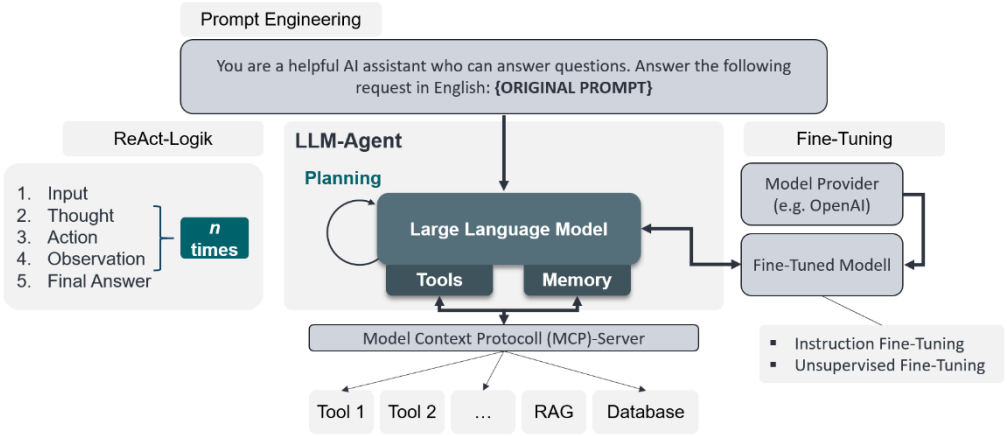

Figure 2: Conceptual description of the internal logic of an LLM agent.

customized prompt instructions for each agent lead to different documents that will be retrieved from the MongoDB server. The implementation of these tools is facilitated by a Modell Context Protocol (MCP) server, which serves as the central registry for all implemented tools. Furthermore, the MCP server's functionality extends to the registration of more complex processes, including information retrieval processes and simulation runs.

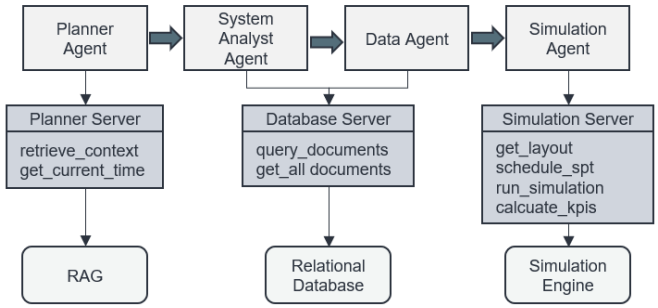

Figure 3: Workflow and available tools and databases for the LLM-agents.

3.2. Digital Twins as Environment

The LLM-agent operates within a digital twin of a production system that has been developed using Siemens' Technomatix Plant Simulation. This discrete-event simulation model comprises objects such as machines, buffers, conveyors, transporters, and workers. Each object incorporates configurable parameters, encompassing cycle times, capacities, routing logic, and maintenance patterns. The simulation provides interfaces for importing parameter sets (e.g., buffer sizes, production schedules) and exporting key performance indicators (e.g., throughput, cycle times, utilization rates). The LLM-agent also accesses a graph-based database containing problem statements (e.g., capacity planning), related experiments (e.g., order sequencing), and optimization objectives (e.g., throughput maximization). The database also includes documentation that describes both the simulation model and the real system. This integrated environment enables the LLM-agent to iteratively analyze, plan, and optimize production scenarios. To do this, the agent calls via the COM interface of Plant Simulation use case specific upload functions to load data and to receive the output KPIs.

3.3. Interaction between Agents

This section is the core of the architectural design for multi-LLM-agent systems, highlighting how control, task delegation, and communication are managed among agents.

3.3.1. Architectural Design

The fundamental process of the multi-LLM-agent architecture is illustrated in Figure 4. A LLM receives a prompt and decides on an action about what to do next. If it includes a tool call, a tool node wrapper handles the tool calls of all agents to format the input into the necessary structure for the tools. Then the request is forwarded to a tool request mediator, which is responsible to augment these tool requests. The mediator then calls the registered tools of the MCP Server. The requested tools are executed and the result is provided to the LLM agent. This establishes also the ReAct logic, as the agent is able to act, observe and conclude from the observation about next steps. However, if no tool call was necessary a handover to another agent is performed and the fundamental process restarts.

We have designed the architecture to follow the workflow of a simulation study following the VDI 3633. Consequently, we have developed a Planner Agent to understand the task and plan the tasks, a System Analyst that evaluates the production system and its elements and provides information, a Data Agent that procures and prepares the data for the simulation, and a Simulation Agent that executes simulation runs. In this Architecture, the sequence of agents and thereby the sequence of addressing the user query is predetermined, and at no point can any agent independently decide which agent to transfer the control to (this applies to our setup, although not necessarily in general; alternative configurations may allow branching workflows), as shown in Figure 3. If an agent has not received a task from the Planner Agent, it logs a 'pass' and contacts the next agent directly. While this setup is less flexible, it offers greater predictability, as it reduces potential communication errors.

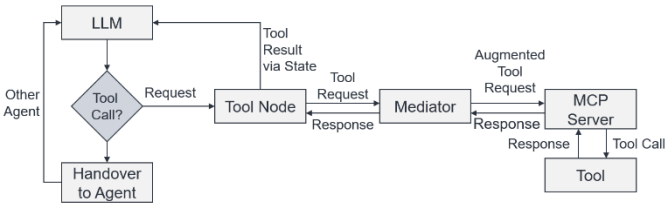

Figure 4: The fundamental process of the workflow architecture.

3.3.2. Communication Protocols

To allow tool calling and the exchange of information between agents, the design process must encompass the establishment of protocols. In open architectures, it is essential to ensure that all agents can communicate with multiple agents. Furthermore, it is crucial to ensure that only the necessary information is transferred to the relevant agent.

The Model Context Protocol (MCP) standardizes communication between agents and their tools using a client–server model with JSON-RPC 2.0 over STDIO (local) or HTTP (remote). This allows a simplified sharing of tool sets, concurrent access to the tools by multiple agents, and the automated exchange of capabilities between server and client.

The Agent-to-Agent Protocol (A2A) serves to complement the MCP by standardizing communication between agents. It is founded on JSON-RPC 2.0 over HTTP, facilitating the exchange of Agent Cards that delineate their respective capabilities. This fosters interoperability within supervisor and network architectures, thereby enabling collaboration across diverse models and providers.

In our setup, MCP is used to facilitate access to tools for each agent. At the same time, A2A is designed to enable inter-agent communication within each architecture.

4. Case Study of the Multi-LLM-Agent Architecture

In this section we will introduce a case study to evaluate these different multi-LLM-agent architectures. Subsection 4.1, provides a description of the case study, while Subsection 4.2

offers a comprehensive explanation and evaluation of the results, which will be further discussed in Subsection 4.3.

4.1. Case Study: Understanding the Ramp-down of a Production Line

The case study examines a manufacturing system characterized by a high degree of product variety. In this case study the company is facing significant fluctuations in customer demand with short-term changes that affect, the productivity of the production system. This necessitates operational adjustments to production planning, including the rescheduling of orders, adaptation of shift patterns, and optimization of inventory levels. The operational adjustments represent a complex planning environment, which is often due to its complexity not fully understood by those responsible for value streams and is therefore not solved optimally.

For each of the challenges, established procedures already exist that are suitable for developing solutions, considering the secondary and boundary conditions. Here, material flow simulation studies are particularly well suited to addressing these challenges, as they enable various solution concepts to be evaluated quickly. Therefore, VDI 3633 [22] is the baseline against which the suitability of the developed multi-LLM-agent architectures in conducting material flow simulation studies with the Software Technomatix Plant Simulation is evaluated.

4.2. Evaluating the Multi-LLM-Agent Workflow Architecture

To test the above described multi-LLM-agent-based architecture, we implemented the proposed workflow architecture using the LLM model Gemini 2.5 Flash for all agents. A questionnaire with 30 questions of varying complexity (an excerpt is shown in Table 1) has been prepared, to evaluate the developed architecture.

Table 1. Excerpt of the question catalog with expected tools calls of the agent.

Question	Correct Tool
Question 1: Is the bike production line a flowshop?	RAG
Question 2: How many orders are there waiting for the car production line as of now?	SQL Query
Question 3: In what sequence should the new jobs (as of 2025-06-19T08:00:00Z) of the Custom Bike Assembly Line be processed to minimize total completion time?	Schedule_spt

To run the experiments, the system is reinitialised for each question, thereby ensuring that the experiments are independent of each other. A single agent architecture is utilized as a baseline. It is based on the same LLM and receives a specific prompt including all tools and steps that are also described to all agents of the workflow architecture.

The results of our experiments are provided in Table 2. In general, the experiments show that the task completion rate (successfully answered questions) is significantly higher for the workflow architecture compared to the single agent. However, the single agent is 32% faster in answering all questions, which can be explained by the inter-agent communication and the coordination effort of the planner agent.

Table 2. Results of the experiment based on task completion rate, used tokens and duration of the experiment.

Architecture	Task Completion Rate	Input tokens	Output Tokens	Duration
Single Agent	50.00 %	242,437	**4,018**	**108.62s**
Multi Agent (Workflow)	**83.33 %**	273,854	9,662	159.75s

Looking at the workflow architecture in detail, the experiments required a total of 283,516 tokens. Of these, 273.854 tokens were attributed to input, while only 9,662 tokens corresponded to output. Based on the pricing of Gemini-2.5-Flash (USD 0.30 per million input tokens and USD 2.50 per million output tokens), the estimated usage cost amounted to approx. USD 0.0028 per question. For each agent at a later stage in the workflow, the quantity of input tokens increases. The planner agent is responsible for 5%, the system analyst for 25%, the data agent for 33% and the simulation agent for 37% of the input tokens, owing to the necessity of storing a greater amount of information within the state description. The data agent is responsible for the provision of descriptions of processes and the planner agent describes the plan and allocates the tasks which are more extensive than the results of the other agents, and as such have the most output tokens (3,292 tokens and 2,948 tokens). In comparison, the single agent required for all experiments combined 11.47% less tokens. For the single agent, the estimated usage cost amounted to approx. USD 0.0027 per question. Evaluating the token usage and cost of running these architectures, shows that having multiple LLM-agents leads to a larger amount of output tokens as each agent uses output tokens to forward the task to the next agent. However, as the costs are calculated per million tokens, the actual cost difference is only USD 0,0001411.

4.3. Discussion

With regard to the failed completion of the tasks, it was frequently observed that the agent's behavior would extend beyond the scope of its assigned tasks, often attempting to address subtasks for which it was not intended and lacked the necessary tools. This might be due to the single reasoning approach, the overload of task complexity, a loss of task focus or the propagation of early errors. Also, the sequential, non-branching workflow limits tasks that require different agents based on a pre-processor's decision. Introducing OR-branches would allow the workflow to route tasks dynamically.

The cost of the more complex architecture in terms of speed is likely acceptable for batch processing of tasks, however it might be a factor in single real-time scenarios. Though, on average, the more complex workflow architecture is only two seconds slower compared to the single agent. Furthermore, the importance of prompt engineering was demonstrated, as agents who explicitly stated that they had completed their task caused subsequent agents to assume that the overall task was finished as well, leading them to prematurely pass their own tasks.

In addition to prompt design, it is crucial to emphasize that the transfer of state descriptions to subsequent agents has a significant impact on the scope and quality of the responses and the feasibility of the system. The high output range of the data

agent and the associated extensive input token required for the simulation agent show that a clever design of the state and the transfer of only relevant information are of central importance for multi-agent architectures.

5. Conclusion and Outlook

We have developed a workflow based multi-LLM-Agent based architecture that demonstrated that it is able to answer significantly more questions correctly compared to a single agent system. This demonstrates that more complex and extensive questions require new approaches and, consequently, new architectures to provide users with correct answers, notably when the range of questions varies considerably.

Future directions emerge from the implications of these findings. We will develop further questions of various complexity classes for the question catalog, which will provide further insights into the behavior of the multi-LLM-agent architecture. Furthermore, we will develop additional architectures to evaluate the suitability of these depending on questions and use cases. Since the choice of model will determine the operational cost for companies, we would also like to focus future research on the usage of small language models (SLMs), as they require significantly lower computational resources, enabling deployment on edge devices with limited memory and energy capacity.

Acknowledgements

This research was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - LA 2351/88-1.

References

[1] Ferreira, C., Armellini, F., & Santa-Eulalia, L. A. (2020). Simulation in Industry 4.0: A state-of-the-art review. Computers & Industrial Engineering, 149, 106868.

[2] Collins, A., Accorsi, R., Mogale, D., et al. (2023). Ten simple rules for credible model verification and validation. Journal of Defense Modeling and Simulation, 20(4), 361–371.

[3] Robinson, S., Monks, T., & Tako, A. A. (2024). Assumptions and simplifications in discrete-event simulation modelling. Journal of Simulation.

[4] Lazarova-Molnar, S., Lechevalier, D., Cassandras, C. G., et al. (2024). Validation of digital twins in labor-intensive manufacturing: Significance and challenges. Procedia CIRP (Elsevier), in press.

[5] Jackson, I., Saenz, M. J., & Ivanov, D. (2024). From natural language to simulations: Applying AI to automate simulation modelling of logistics systems. International Journal of Production Research, 62(4), 1434–1457.

[6] Yao, S., Zhao, J., Yu, D., et al. (2022). ReAct: Synergizing reasoning and acting in language models. arXiv:2210.03629.

[7] dos Santos, C. H., Montevechi, J. A. B., de Queiroz, J. A., de Carvalho Miranda, R., & Leal, F. (2021). Decision support in productive processes through DES and ABS in the Digital Twin era: a systematic literature review. International Journal of Production Research, 60(8), 2662–2681. https://doi.org/10.1080/00207543.2021.1898691

[8] VDI 3633 Blatt 1. (2014). *Simulation von Logistik-, Materialfluss- und Produktionssystemen – Grundlagen* [Simulation of systems in materials handling, logistics and production – Fundamentals]. Beuth Verlag. https://www.dinmedia.de/de/technische-regel/vdi-3633-blatt-1/149034959

[9] Overbeck, L., Graves, S. C., & Lanza, G. (2023). Development and analysis of digital twins of production systems. International Journal of Production Research, 62(10), 3544–3558. https://doi.org/10.1080/00207543.2023.2242525

[10] Mertens, J., & Denil, J. (2023). Digital-twin co-evolution using continuous validation. In Proceedings of the ACM/IEEE 14th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2023) (pp. 266–267). Association for Computing Machinery. https://doi.org/10.1145/3576841.3589628

[11] May, M. C., Overbeck, L., Wurster, M., Kuhnle, A., & Lanza, G. (2021). Foresighted digital twin for situational agent selection in production control. *Procedia CIRP, 99, 27–32. Elsevier. https://doi.org/10.1016/j.procir.2021.03.005*

[12] Villalonga, A., Negri, E., Biscardo, G., Castano, F., Haber, R. E., Fumagalli, L., & Macchi, M. (2021). A decision-making framework for dynamic scheduling of cyber-physical production systems based on digital twins. *Annual Reviews in Control*, *51*, 357-373.

[13] Ansari, F. (2024). Wissensmanagement in der Industrie 4.0. In R. Schulte, C. M. Schmitz, & H. Jodlbauer (Eds.), *Handbuch Unternehmensorganisation: Strategien, Planung, Umsetzung* (pp. 1–15). Springer Vieweg. https://doi.org/10.1007/978-3-642-45370-0_97-1

[14] Brundage, M. P., Sexton, T., Hodkiewicz, M., Dima, A., & Lukens, S. (2021). Technical language processing: Unlocking maintenance knowledge. *Manufacturing Letters, 27,* 42–46. https://doi.org/10.1016/j.mfglet.2020.11.001

[15] May, M. C., Neidhöfer, J., Körner, T., Schäfer, L., & Lanza, G. (2022). Applying natural language processing in manufacturing. *Procedia CIRP, 115,* 184–189. https://doi.org/10.1016/j.procir.2022.10.071

[16] Rane, N., Choudhary, S., & Rane, J. (2024). Intelligent manufacturing through generative artificial intelligence, such as ChatGPT or Bard. *SSRN.* https://doi.org/10.2139/ssrn.4681747

[17] Russell, S. J., & Norvig, P. (1995). *Artificial intelligence: A modern approach*. Prentice Hall.

[18] Weiss, G. (Ed.). (1999). *Multiagent systems: A modern approach to distributed artificial intelligence*. MIT Press.

[19] Zhao, Z., Tang, D., Zhu, H., Zhang, Z., Chen, K., Liu, C., & Ji, Y. (2024). A large language model-based multi-agent manufacturing system for intelligent shopfloor. arXiv preprint arXiv:2405.16887. Retrieved from https://arxiv.org/abs/2405.16887

[20] Lim, J., Vogel-Heuser, B., & Kovalenko, I. (2024). Large language model-enabled multi-agent manufacturing systems. Proceedings of the 2024 IEEE 20th International Conference on Automation Science and Engineering (CASE). IEEE. https://doi.org/10.1109/CASE53703.2024.10094128

[21] Xia, Y., Shenoy, M., Jazdi, N., Weyrich, M., Zhang, J., Shah, C., & Ji, Y. (2023). Towards autonomous system: Flexible modular production system enhanced with large language model agents. Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation (ETFA). IEEE. https://doi.org/10.1109/ETFA52470.2023.10112415

[22] Chen, Y., Arkin, J., Zhang, Y., Roy, N., & Fan, C. (2023). Scalable multi-robot collaboration with large language models: Centralized or decentralized systems?. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). IEEE. https://doi.org/10.1109/ICRA48891.2023.10112415

[23] Yang, T., Feng, P., Guo, Q., Zhang, J., Zhang, X., & Liu, Y. (2025). AutoHMA-LLM: Efficient task coordination and execution in heterogeneous multi-agent systems using hybrid large language models. IEEE Transactions on Cognitive Communications and Networking. https://doi.org/10.1109/TCCN.2025.10094128

[24] Vyas, J., & Mercangöz, M. (2024). Autonomous industrial control using an agentic framework with large language models. arXiv preprint arXiv:2411.05904. Retrieved from https://arxiv.org/abs/2411.05904

[25] Lewis, P., Perez, E., Piktus, A., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems, 33, 9459–9474.

[26] Yao, S., Zhao, J., Yu, D., et al. (2022). ReAct: Synergizing Reasoning and Acting in Language Models. arXiv preprint arXiv:2210.03629.

[27] Chen, Y., Zhang, K., Wang, W., & Wang, L. (2025). Unleashing the potential of prompt engineering for large language models: A comprehensive review. *Patterns, 6*(6), 101260. https://doi.org/10.1016/j.patter.2025.101260