Large Language Model-Enabled Multi-Agent Manufacturing Systems

Jonghan Lim[1], Birgit Vogel-Heuser[2], and Ilya Kovalenko[1]

Abstract— **Traditional manufacturing faces challenges adapting to dynamic environments and quickly responding to manufacturing changes. The use of multi-agent systems has improved adaptability and coordination but requires further advancements in rapid human instruction comprehension, operational adaptability, and coordination through natural language integration. Large language models like GPT-3.5 and GPT-4 enhance multi-agent manufacturing systems by enabling agents to communicate in natural language and interpret human instructions for decision-making. This research introduces a novel framework where large language models enhance the capabilities of agents in manufacturing, making them more adaptable, and capable of processing context-specific instructions. A case study demonstrates the practical application of this framework, showing how agents can effectively communicate, understand tasks, and execute manufacturing processes, including precise G-code allocation among agents. The findings highlight the importance of continuous large language model integration into multi-agent manufacturing systems and the development of sophisticated agent communication protocols for a more flexible manufacturing system.**

I. INTRODUCTION

The escalating demand for customized products and the increasing complexities of production processes are driving a transformation in the manufacturing sector. This paradigm shift has required manufacturers to quickly adjust to evolving product specifications and adeptly address unforeseen operational changes. In this context, manufacturers are required to not only maintain high-quality products and cost efficiency but also be agile and responsive to a dynamically changing environment.

One significant approach for enhancing adaptability and quick response of manufacturing systems is using multi-agent control [1]. A Multi-Agent System (MAS) includes several autonomous agents for decision-making, communication, and coordination [2]. Many multi-agent architectures have been developed for manufacturing systems [3], [4]. By enabling decentralized decision-making and enhancing coordination among various agents, MAS have contributed significantly to handling complexities in the manufacturing system [5].

However, there are still challenges in MAS frameworks in manufacturing. While agents in MAS can automate processes and make decisions, their ability to adapt to new formats or specifications is limited, leading to increased downtime.

Thus, a challenge includes the limited ability of agents to dynamically adapt to new information including product specifications *(RCh1)*. Another challenge lies in processing unstructured textual data. Textual data is crucial because it is more expressive than numerical data, offering vast knowledge that is currently underutilized in manufacturing, where structured data fails to include varied influencing factors [6]. Natural language processing capabilities to formalize unstructured textual data can reduce downtime and optimize processes [6]. Therefore, natural language processing is essential for interpreting and implementing textual data from human operators or interconnected systems *(RCh2)*.

The advent of Large Language Models (LLMs) presents a novel solution to these challenges. LLMs, such as OpenAI's GPT-3 [7] and GPT-4 [8], provides opportunities to bring more flexibility and adaptability. With Machine to Machine (M2M) protocols, such as Message Queuing Telemetry Transport (MQTT) [9] and MTConnect [10] that uses predefined structures, incorporating LLMs offer benefits. LLMs can interpret, generate, and act on natural language instructions, facilitating complex decision-making processes and adaptive responses to changing manufacturing conditions [11]. The integration of natural language processing capabilities may also benefit operators and other system users who have limited experience with the specific process or system [12]. Natural language processing capabilities can allow new users to quickly adapt to managing and optimizing processes and resources [13]. The main contributions of this paper include: Leveraging LLM to allocate and execute manufacturing tasks based on user instructions to enhance task efficiency *(Cn1)*, enabling LLM agents to adapt to evolving specifications ensuring the production process is executed with the appropriate resources *(Cn2)*, and applying natural language processing capabilities to enable agents to comprehend context-specific instructions from users and other agents *(Cn3)*.

The rest of the manuscript is organized as follows. Section II provides background regarding MAS, LLMs in MAS, and LLMs in manufacturing. Section III details the functioning of the proposed framework. Section IV demonstrates a case study to illustrate the applications and effectiveness of our framework. Section V presents challenges and limitations while integrating LLM. Section VI summarizes the paper and discusses future work.

II. BACKGROUND

A. Multi-Agent Systems in Manufacturing

Various MAS architectures have been developed to enhance agent communication, significantly contributing to the

[1]Jonghan Lim and Ilya Kovalenko are with the Department of Industrial and Manufacturing and the Department of Mechanical Engineering, Pennsylvania State University, State College, USA (e-mail: {jxl567; iqk5135}@psu.edu).

[2]Birgit Vogel-Heuser is with the Institute of Automation and Information Systems, Technical University of Munich, Munich, Germany (e-mail: vogel-heuser@tum.de).

increased manufacturing system flexibility. For instance, the Axiomatic Design of a Multi-Agent Reconfigurable Mechatronic System (ADMARMS) framework facilitates design-based reconfigurability [14]. Studies by Wang et al. [15] and Kim et al. [16] integrated MAS with big data analytics and reinforcement learning. Additionally, Kovalenko et al. [17] and Bi et al. [4] focused on dynamic resource and task allocation within MAS, with frameworks for resource task negotiation and a model-based resource agent architecture to address manufacturing disruptions.

While these MAS architectures were developed to enhance agent communication for increased flexibility, they focus on utilizing predefined algorithms for task allocation and agent communication. Furthermore, there has been limited exploration in using natural language interpretation from human operators or other agents to enhance the responsiveness of manufacturing process control. By leveraging LLMs, agents can improve operational flexibility and efficiency in addressing these gaps.

B. Large Language Models in Multi-Agent Systems

The emergence of LLMs has sparked interest in developing interacting autonomous agents powered by the LLM technology [18]. LLM-enabled agents demonstrate enhanced problem-solving capabilities through multi-agent discussions and diverse applications. For instance, Park et al. [19] showcase a simulation with 25 generative agents in a virtual town, facilitating social understanding studies. Another work by Wu et al. [20] introduces an open-source framework called *AutoGen* that facilitates customizable agent interactions and conversation programming. ChatEval [21] uses debater agents in an LLM-powered MAS for text evaluation, achieving enhanced accuracy and depth over traditional methods and single agent setups. MetaGPT [22], a meta-programming framework, enhances multi-agent collaboration using an assembly line approach for software engineering tasks.

These works demonstrate advanced multi-agent interactions, realistic behavior simulation, and task efficiency, which can enhance adaptability, coordination, and advanced natural language processing capabilities in manufacturing. Building on these insights, we explore integrating LLMs in a multi-agent framework to address the unique requirements and benefits in the manufacturing sector.

C. Large Language Models in Manufacturing

With the advancement of LLM technology, recent studies have been increasingly exploring both the opportunities and challenges of implementing LLMs within the manufacturing domain. Makatura et al. [11] explores the application of LLMs in design and manufacturing, including CAD/CAM. Their approach investigates the potential role of LLMs in a variety of tasks, including design specification and manufacturing instruction, and highlights the capabilities and limitations. The analysis of GPT-4 shows strengths in design and manufacturing knowledge but faces challenges in reasoning and accuracy, with potential solutions including

specialized languages and better context management [11]. Jignasu et al. [23] provide a comprehensive evaluation of LLMs for understanding, debugging, and manipulating G-code in 3D printing, testing their capabilities in error detection, correction, and geometric transformations. Similar work by Badini et al. [24] explores using ChatGPT to optimize the G-code generation in additive manufacturing. These studies prove that LLM can interpret G-code, essential for tasks like debugging, manipulating, and comprehending the code. Fakih et al.[25] and Koziolek et al. [26] focus on enhancing programming in industrial control systems, specifically Programmable Logic Controllers (PLCs), with frameworks like LLM4PLC and studies assessing GPT-4's capability to generate control logic from natural language prompts.

Several studies have integrated LLMs into manufacturing, including CAD/CAM, G-code, and PLCs, to enhance efficiency. However, these studies have not focused on using LLMs with MAS for autonomous task allocation, adaptability to evolving specifications, and interpretation of context-specific instructions. The framework described in Section III leverages LLM to enhance multi-agent interactions and decision-making for manufacturing processes. This framework develops autonomous agents that use context-specific natural language instructions to dynamically adjust to new product information and coordinate tasks in real-time. Our approach is an initial step toward enhancing the CAD/CAM workflow, automatically adapting CAM processes based on changes in CAD specifications, ensuring seamless manufacturing operations.

III. FRAMEWORK

The novel architecture in Figure 1 is designed based on a MAS that includes both Product Agents (PAs) and Resource Agents (RAs). A PA is an agent responsible for decision-making related to a product, while an RA serves as a controller for a specific resource, such as a robot or machine [27].

Figure 1 provides an overview of an LLM-enabled multi-agent manufacturing system framework, where users initialize agents with instructions and capabilities. PAs are assigned specific processes including G-code specifications, while possessing the ability to communicate and access historical data for informed decision-making. RAs are responsible for the process execution of tasks, informed by instructions and tool specifications. LLM is applied for processing natural language inputs from both PAs and RAs. For evaluation, the chosen LLM is GPT-4, the latest and most advanced version of GPT developed by OpenAI [8].

This framework facilitates adaptability, enables real-time coordination, and provides the ability to respond to the demands of a complex manufacturing environment with natural language processing capabilities.

A. Agent Initialization

The agent initialization process configures a MAS by defining roles, capabilities, and communication protocols for
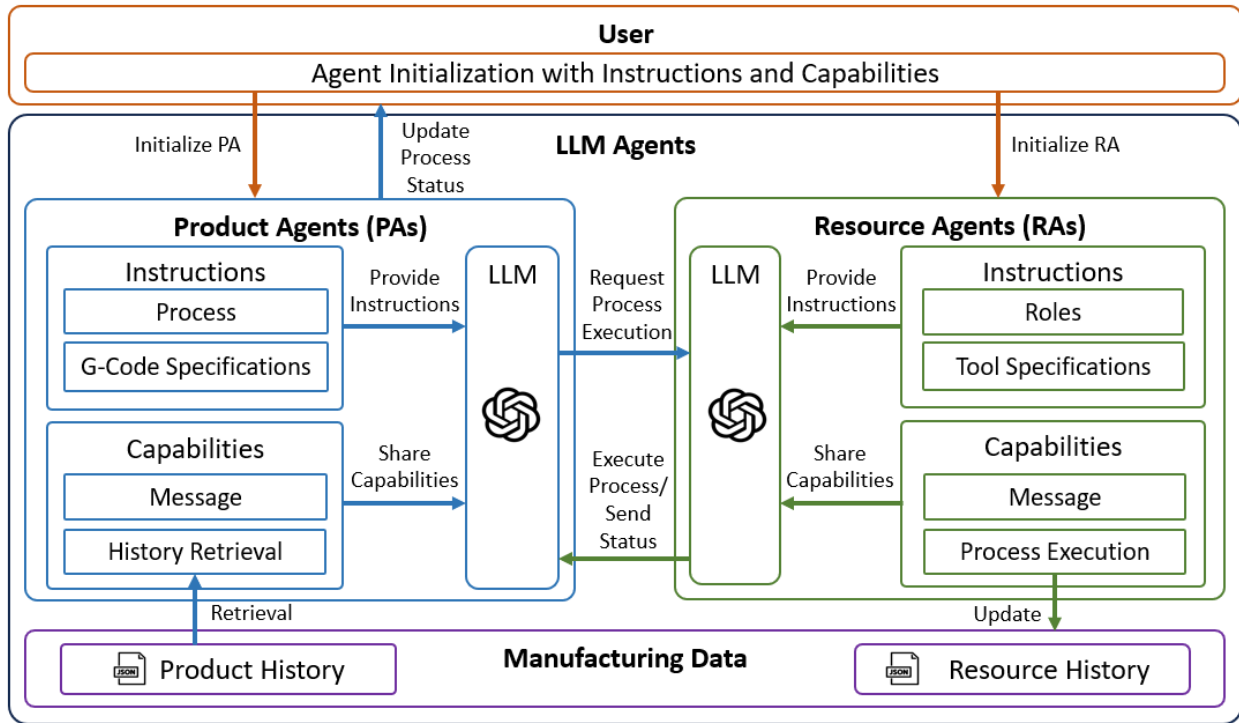
Fig. 1: Framework for LLM-Enabled Multi-Agent Manufacturing Systems

---

Algorithm 1: Agent Creation

1: **procedure** CREATEAGENTS($Agent\_list$, $Agent\_functions$)
2:    $Agents \leftarrow []$
3:    **for** $agent\_initialization$ in $Agent\_list$ **do**
4:      $configuration \leftarrow$ LoadJSON($agent\_initialization$)
5:      **for** $name$, $agent\_data$ in $configuration$ **do**
6:        $function\_list \leftarrow []$
7:        **if** 'functions' in $agent\_data$ **then**
8:          **for** $function\_name$ in $agent\_data$['functions'] **do**
9:            $function\_reference \leftarrow$ GetFunction($Agent\_functions$,
10:            $function\_name$)
11:          **if** $function\_reference \neq$ None **then**
12:            $function\_list$.append($function\_reference$)
13:        $agent \leftarrow$ NewAgent($name$, $agent\_data$, $function\_list$)
14:        $Agents$.append($agent$)
15:    **return** $Agents$

---

each agent. This setup ensures agents operate effectively and collaborate towards system goals. The following initial prompt is given to all agents in the system:

```
"You are a helpful agent in a cooperative
Multi-Agent System. If you are asked for a service
you can provide you should help. If necessary,
you may ask the other agent for clarifying
information. You may communicate with your
peers to achieve your goals. If you do not know
the answer do not make things up. Only use the
functions you have been provided with. However,
you may call these functions recursively."
```

These instructions ensure each agent operates within its capabilities, engages in effective communication, and contributes to the MAS's objectives.

Following the initial prompts, the initialization process assigns specific roles to agents using various parameters.

These parameters are selected to achieve adaptability, real-time response, autonomous decision-making, and efficient coordination within the MAS. Key parameters include:

- **Functions**: Capabilities within the manufacturing that range from retrieving history to executing tasks.
- **Annotation**: Agent's role and responsibilities, which aid in selecting the appropriate LLM agent.
- **Instructions**: User instructions offer a sequence of operations, techniques, and standards for agent activities.
- **Inbox**: The agent receives and manages instructions, updates, and feedback from other agents.

For PAs, an additional parameter is introduced:

- **Specification File**: Detailed manufacturing specifications, such as G-code files for CNC operations, for precise manufacturing execution.

RAs are further configured with:

- **Configurations**: This parameter outlines the operational settings specific to RAs, such as process times, defect rates, and breakdown probabilities.

Integrating parameters into LLM-enabled MAS for manufacturing extends standardized approaches (e.g. FIPA [28] and JADE [29]) by incorporating natural language processing for more flexible agent interactions. Integrating user-defined instructions and parameters in agent initialization can be further expanded through user studies to ensure industry applicability.

Based on the initialization information, the Agent Creation function, cp. Algorithm 1, dynamically generates agents. The Agent Creation function provides advanced LLM agents with different capabilities in manufacturing

Algorithm 2: Chat Interaction

```
1: procedure CHAT(prompt, model)
2:    with_functions ← len(function_info) > 0
3:    msgs ← []
4:    if instructions then
5:       msgs.append({"role": "system", "content": instructions})
6:    msgs.append({"role": "user", "content": prompt})
7:    response ← __get_response(msgs, model_, with_functions)
8:    msgs.append(response.message)
9:    function ← response.message.function_call.name
10:   args ← json.loads(response.message.function_call.arguments)
11:   function_response ← executables[function](**args)
12:   return function_response, msgs
```

systems for adaptability to changing requirements. The algorithm iterates over each entry in the *Agent_list*, which contains initialization information. For every entry, configuration data *configuration* for the agent is loaded from a JSON, containing details such as the agent's name, role, capabilities, and any specific functions. For each agent, an empty list *function_list* is prepared. If the *'functions'* key is present, the algorithm iterates through each function name listed. For each function name, a reference *function_reference* is obtained by searching within a provided set of agent functions *Agent_functions*. If the function is found, its reference is appended to *function_list*. A new agent instance *agent* is created with the specified name, agent data, and the list of function references. This agent is then added to the *Agents* list. This algorithm facilitates the flexible creation of agents allowing for adaptation to various tasks.

B. Agent Capabilities and LLM Integration

Through the integration of LLM, agents gain the capability to interpret complex instructions and specifications, offering a flexible approach to manufacturing operations. The framework uses LLM to decode user instructions for manufacturing and leverages the OpenAI Application Programming Interface (API) feature called *function calling* [30]. This feature allows for the automatic execution of user-defined functions, including specific manufacturing operations (e.g., milling, drilling) through descriptions that include operation parameters and required inputs like process name and product specifications (e.g. G-code).

The `'chat'` function, cp. Algorithm 2, facilitates interaction among agents, based on available function information and executing function calls as defined by the manufacturing process requirements. The outcome, *function_response*, includes the details like operation type, product, and machine names, along with a message, *msgs*, to facilitate the subsequent steps in the process. This approach enables context-aware execution of tasks, enhancing adaptability and efficiency.

C. Operational Workflow

The workflow, as depicted in Figure 2, shows how user instructions in textual data can be converted into actionable operations. This approach enables the translation of high-level directives of users into machine-executable actions, enhancing the responsiveness to changing conditions.
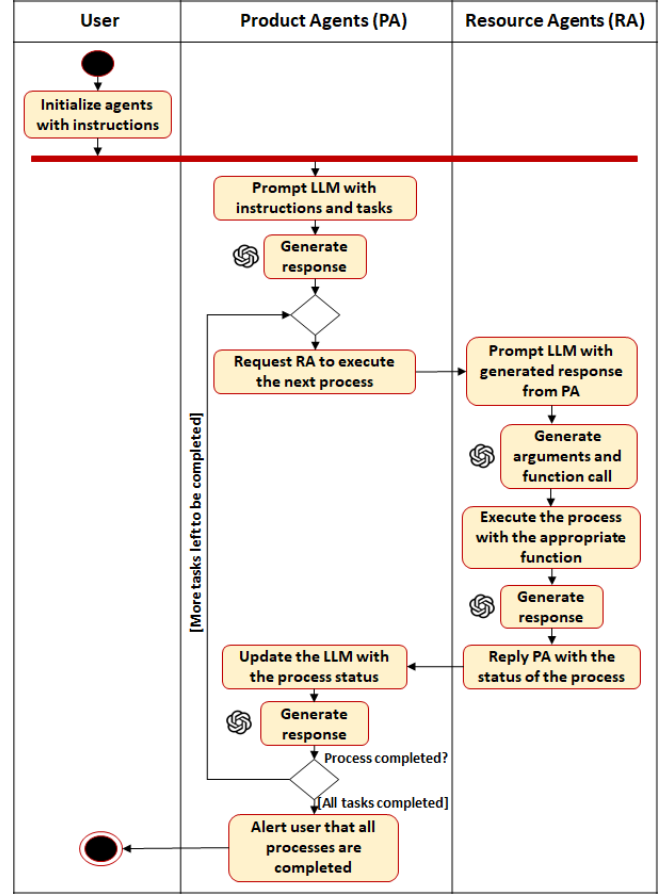

Fig. 2: Operational Workflow of the Framework Using LLM for Task Execution and Coordination

The workflow begins with the user initializing PAs and RAs. Users describe product goals, sequences of operations, and specifications. This information is then processed by the LLM, which generates responses to guide the process. The PA determines the suitable RA for the next task, prompting it to carry out the required operation. The selected RA constructs the necessary arguments to request the corresponding function call, leading to the execution of the specified process. Upon successful operation, the LLM formulates an appropriate response, informing the PA of the process status. This feedback updates the LLM-integrated PA with the current status, determining the completion of the task. If additional tasks remain, the cycle repeats; otherwise, the user is notified of the completion of all processes.

IV. CASE STUDY

In this case study, we examine the proposed framework and demonstrate how the LLM is integrated into the framework to execute the manufacturing process.

A. Case Study Setup

A simulated manufacturing environment includes three machines specialized for milling, drilling, and threading tasks, as highlighted in Figure 3. To evaluate the system's adaptability, we structured the case study into two distinct
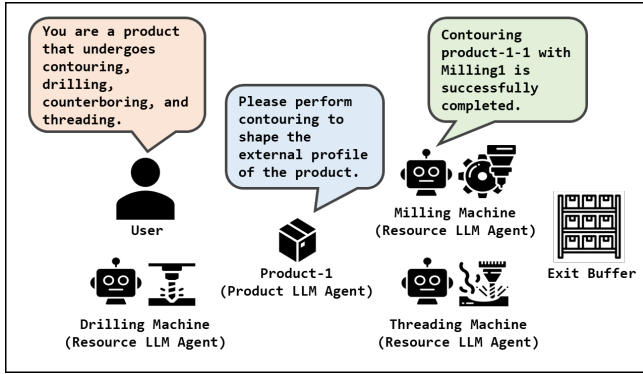
Fig. 3: Case Study Setup

Fig. 4: Case Study G-code Product Specification

processes: a 2-step process and a more comprehensive 4-step process. In the 2-step process, a prototype product, referred to as 'product-1', is subjected to contouring and drilling operations. The 4-step process adds counterboring and threading operations, testing the system's capability to handle extended workflows. The G-code specifications are adapted from [31], as detailed in Figure 4. Notably, explicit labels for G-code specifications (e.g. Step1-Contouring, Step2-Drilling) are not provided to the agent. Leveraging LLM's capability to interpret and assign appropriate G-code based on tool specifications for each RA demonstrates the system's potential to manage G-code assignments, demonstrating the capabilities of LLMs.

Each product has an LLM-enabled PA which oversees high-level decision-making for its associated product. For instance, 'product-1' is instructed for a 2-step process with the following instructions:

```
"Make sure to get contouring to shape the
external profile of the product. Then, get a
drilling operation to create precise holes.
Proceed to Exit Buffer when completed. Your
product name is 'product-1'. The G-code
specifications are as follows: %358 N1 N10 G90
G71 G80 G40..."
```

Each LLM-enabled RA handles decisions for its specific machine. 'Milling1' is tasked with both contouring and counterboring to demonstrate how LLM can assign operations adaptively. 'Drilling1' focuses on drilling, while 'Threading1' manages threading. Machines are assigned tool numbers used in G-code to ensure PAs provide relevant G-code to each RA. For example, the 'Milling1' machine receives this operational prompt:

```
"Perform milling operations as required,
including contouring the external shape and
counterboring for holes. Your machine name is
Milling1. The tool number for contouring is T4.
The tool number for counterboring is T6."
```

This experiment investigates the LLM's capability to identify and allocate correct manufacturing task and their corresponding G-code specifications. It aims to enhance the adaptability by showcasing natural language communication between PA and RA, simulating human-like interaction for task execution.
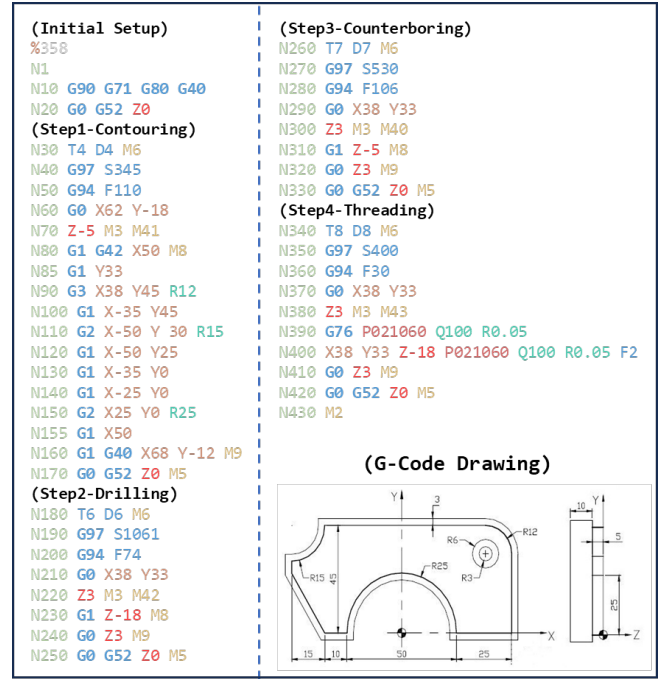
B. Case Study Simulation and Results Analysis

Following the case study setup, simulations were conducted to evaluate the system, as shown in Figure 5. In this example, the user initiates the workflow by instructing the 'Product-1' agent and all RAs, using Algorithm 1 from Section III-A. After initialization, the system follows the operational workflow in Section III-C. The 'Product-1' agent first communicates the need for contouring to the 'Milling1' agent, providing only the contouring G-code section. After the 'Milling1' agent completes the task, it informs the 'Product-1' agent, indicating it is ready for the next operation. The process continues with the 'Product-1' agent assigning the drilling operation to the 'Drilling1' agent, which updates the 'Product-1' agent after completion. When all operations are completed, the 'Product-1' agent updates the user, indicating it is ready to move to the exit buffer. All agent communications are facilitated by the chat interaction protocol defined in Algorithm 2, as outlined in Section III-B.

1) Communication and Comprehension: Our case study evaluated the agents' ability to communicate and comprehend specific tasks. Using natural language processing, agents interacted in both 2-step and 4-step processes. In the 2-step process, the communication between the 'Product-1' agent and the 'Milling1' and 'Drilling1' agents showed the system's efficiency in handling straightforward task sequences and distributing G-code specifications. The 4-step process tested the agents' ability to manage complex sequences, with 'Milling1' handling both contouring and counterboring. In this scenario, the 'Product-1' agent coordinated contouring, drilling, counterboring, and threading, showing the agents' ability to interpret complex instructions.

2) Accuracy and Performance: The data in Table I shows GPT-4 achieved 100% success in 2-step and 86% in 4-step
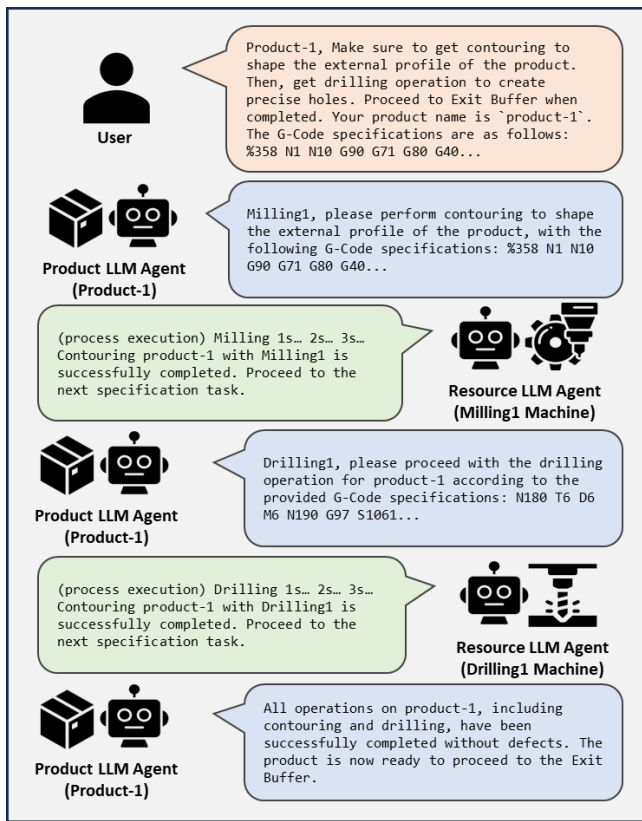
Fig. 5: Agent Communication Result for 2-step Process

TABLE I: Success Rates for Case Studies with GPT-4

Metric	2-Step	4-Step
Success Rate	1	.86
Overall Error Rate	0	.14
- P1. Contouring Failure Rate	0	0
- P2. Drilling Failure Rate	0	.06
- P3. Counterboring Failure Rate	N/A	.02
- P4. Threading Failure Rate	N/A	.06

TABLE II: Detailed Error Analysis with GPT-4

Error Type	2-Step	4-Step
Incorrect Function Calls	0	.43
Inaccurate G-code Allocation	0	.14
Process Inexecution	0	.43

tion. While effective in processing natural language, LLMs sometimes lack precision, as indicated in Section IV-B.2. This issue raises concerns about the feasibility of relying on LLMs for tasks where errors can cause machine breakdown and low product quality. The proposed framework can enhance performance by applying fine-tuned LLMs on domain-specific data [32].

B. Scalability

As data volume and task complexity grow, LLMs may struggle to maintain accuracy. This limitation is particularly noticeable in situations where extensive historical data and detailed prompts are essential, as exemplified in the 4-step process in Section IV. The potential decrease in accuracy with larger datasets poses a challenge to scaling LLM-enabled multi-agent control across more complex manufacturing operations. A knowledge base for multi-agent manufacturing systems [33] should be integrated for practical industrial applications to provide LLMs with accurate runtime data and to enhance scalability.

C. Reliability

Reliability is key for integrating this method into real manufacturing scenarios. LLMs generating hallucinated responses or making false decisions raises safety concerns, as shown in Section IV. The LLM allocated inaccurate G-code or executed the incorrect function calls. Moreover, the reasoning provided by LLMs may require validation in practical manufacturing scenarios even if it seems logical. Therefore, it is essential to implement robust verification mechanisms, as well as incorporate real-time observability and monitoring capabilities. This approach ensures LLMs' outputs and reasoning are consistent and allow immediate adjustments with manufacturing standards and safety protocols. Joublin et al. [34] demonstrate a system for robotics that uses LLMs for task planning and replanning in response to errors, highlighting the importance of feedback loops for safety and reliability in complex manufacturing environments.

processes across 50 trials each, with performance declining as task complexity increased. The 4-step process had a 14% overall error rate. The detailed error breakdown in Table II shows that in the 4-step process, 43% of errors were due to incorrect function calls and process failures, and 14% were due to inaccurate G-code allocation. The increase in prompt length correlates with a rise in errors, highlighting the need for model training and improved prompt engineering. Addressing these issues will enhance task execution capabilities in MASs for manufacturing, improving system adaptability and efficiency even in unexpected scenarios.

3) Insights from Case Studies: The case study provided insights into the capabilities of LLM-enabled MASs in manufacturing. The framework uses LLM to understand manufacturing tasks *(Cn1)*, adapt to new product specifications *(Cn2)*, and facilitate agent communication via natural language *(Cn3)*. This enhances system flexibility and task management without relying on predetermined rules. However, accuracy issues with GPT-4 highlight the need for model improvement to manufacturing needs. Difficulties with incorrect function calls point to the necessity for better error management and validation. These results guide future research toward enhancing LLM integration, developing advanced agent communication protocols, and increasing system robustness.

V. CHALLENGES AND LIMITATIONS

A. Performance

A significant challenge in manufacturing using LLMs is achieving the performance required for high-quality produc-

VI. CONCLUSION AND FUTURE WORK

In our research, we explored integrating LLM within MAS to enhance manufacturing processes. Our research demonstrates that LLM improves task allocation efficiency

and operational adaptability *(Cn1)*, enables agents to adapt to evolving product specifications (e.g., G-code) *(Cn2)*, and enhances agent communication and collaboration through advanced natural language processing *(Cn3)*. However, challenges like task execution performance, scalability, and reliability highlight the need for verification and monitoring. Additionally, the diversity in user instructions requires a strategy to ensure reliable task interpretation and execution.

Future work will focus on developing LLM-based communication protocols for agents to handle diverse manufacturing instructions and operator proficiency levels. This approach seeks to showcase the agents' capability to learn and adapt from historical data, enhancing their responsiveness to rapid industry changes. Additionally, efforts will be directed towards enhancing the accuracy and scalability of LLM. While this paper focused on common manufacturing processes (e.g. drilling, milling), future research will include less common processes with larger datasets to evaluate the framework's effectiveness. The goal is to ensure the efficient processing of diverse manufacturing tasks without impacting performance.

REFERENCES

[1] B. Vogel-Heuser, J. Lee, and P. Leitão, "Agents Enabling Cyber-Physical Production Systems," *at-Automatisierungstechnik*, vol. 63, no. 10, pp. 777–789, 2015.

[2] M. Wooldridge, "An Introduction to MultiAgent Systems," 2009.

[3] I. Kovalenko, D. Tilbury, and K. Barton, "The Model-based Product Agent: A Control Oriented Architecture for Intelligent Products in Multi-agent Manufacturing Systems," *Control Engineering Practice*, vol. 86, pp. 105–117, 2019.

[4] M. Bi, I. Kovalenko, D. M. Tilbury, and K. Barton, "Dynamic Distributed Decision-Making for Resilient Resource Reallocation in Disrupted Manufacturing Systems," *International Journal of Production Research*, pp. 1–21, 2023.

[5] I. Kovalenko, E. C. Balta, D. M. Tilbury, and K. Barton, "Cooperative Product Agents to Improve Manufacturing System Flexibility: A Model-based Decision Framework," *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 1, pp. 440–457, 2022.

[6] M. C. May, J. Neidhöfer, T. Körner, L. Schäfer, and G. Lanza, "Applying Natural Language Processing in Manufacturing," *Procedia CIRP*, vol. 115, pp. 184–189, 2022.

[7] L. Floridi and M. Chiriatti, "GPT-3: Its Nature, Scope, Limits, and Consequences," *Minds and Machines*, vol. 30, pp. 681–694, 2020.

[8] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "GPT-4 Technical Report," *arXiv preprint arXiv:2303.08774*, 2023.

[9] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "MQTT-S—A Publish/Subscribe Protocol for Wireless Sensor Networks," *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08)*, pp. 791–798, 2008.

[10] MTConnect, 2008. [Online]. Available: http://www.mtconnect.org/

[11] L. Makatura, M. Foshey, B. Wang, F. HähnLein, P. Ma, B. Deng, M. Tjandrasuwita, A. Spielberg, C. E. Owens, P. Y. Chen, A. Zhao, A. Zhu, W. J. Norton, E. Gu, J. Jacob, Y. Li, A. Schulz, and W. Matusik, "How Can Large Language Models Help Humans in Design and Manufacturing?" *arXiv preprint arXiv:2307.14377*, 2023.

[12] J. Chen, X. Lu, M. Rejtig, D. Du, R. Bagley, M. S. Horn, and U. J. Wilensky, "Learning Agent-based Modeling with LLM Companions: Experiences of Novices and Experts Using ChatGPT & NetLogo Chat," *arXiv preprint arXiv:2401.17163*, 2024.

[13] T. Schulte, G. Mouratidis, and V. Pfanschilling, "From Expert to Novice: Increasing Reactive Resilience by Transfering Knowledge," *Proceedings of the 2021 Pre-ICIS SIGDSA Symposium. 15*, 2021.

[14] A. M. Farid and L. Ribeiro, "An Axiomatic Design of a Multiagent Reconfigurable Mechatronic System Architecture," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 5, pp. 1142–1155, 2015.

[15] S. Wang, J. Wan, D. Zhang, D. Li, and C. Zhang, "Towards Smart Factory for Industry 4.0: A Self-Organized Multi-Agent System with Big Data Based Feedback and Coordination," *Computer networks*, vol. 101, pp. 158–168, 2016.

[16] Y. G. Kim, S. Lee, J. Son, H. Bae, and B. Do Chung, "Multi-Agent System and Reinforcement Learning Approach for Distributed Intelligence in a Flexible Smart Manufacturing System," *Journal of Manufacturing Systems*, vol. 57, pp. 440–450, 2020.

[17] I. Kovalenko, D. Ryashentseva, B. Vogel-Heuser, D. Tilbury, and K. Barton, "Dynamic Resource Task Negotiation to Enable Product Agent Exploration in Multi-Agent Manufacturing Systems," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2854–2861, 2019.

[18] L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, W. X. Zhao, Z. Wei, and J.-R. Wen, "A Survey on Large Language Model Based Autonomous Agents," *arXiv preprint arXiv:2308.11432*, 2023.

[19] J. S. Park, J. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative Agents: Interactive Simulacra of Human Behavior," *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pp. 1–22, 2023.

[20] Q. Wu, G. Bansal, J. Zhang, Y. Wu, S. Zhang, E. Zhu, B. Li, L. Jiang, X. Zhang, and C. Wang, "AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation Framework," *arXiv preprint arXiv:2308.08155*, 2023.

[21] C.-M. Chan, W. Chen, Y. Su, J. Yu, W. Xue, S. Zhang, J. Fu, and Z. Liu, "ChatEval: Towards Better LLM-based Evaluators Through Multi-Agent Debate," *arXiv preprint arXiv:2308.07201*, 2023.

[22] S. Hong, M. Zhuge, J. Chen, X. Zheng, Y. Cheng, C. Zhang, J. Wang, Z. Wang, S. K. S. Yau, Z. Lin, L. Zhou, C. Ran, L. Xiao, C. Wu, and J. Schmidhuber, "MetaGPT: Meta Programming for Multi-Agent Collaborative Framework," *arXiv preprint arXiv:2308.00352*, 2023.

[23] A. Jignasu, K. Marshall, B. Ganapathysubramanian, A. Balu, C. Hegde, and A. Krishnamurthy, "Towards Foundational AI Models for Additive Manufacturing: Language Models for G-code Debugging, Manipulation, and Comprehension," *arXiv preprint arXiv:2309.02465*, 2023.

[24] S. Badini, S. Regondi, E. Frontoni, and R. Pugliese, "Assessing the Capabilities of ChatGPT to Improve Additive Manufacturing Troubleshooting," *Advanced Industrial and Engineering Polymer Research*, 2023.

[25] M. Fakih, R. Dharmaji, Y. Moghaddas, G. Q. Araya, O. Ogundare, and M. A. A. Faruque, "LLM4PLC: Harnessing Large Language Models for Verifiable Programming of PLCs in Industrial Control Systems," *arXiv preprint arXiv:2401.05443*, 2024.

[26] H. Koziolek, S. Gruener, and V. Ashiwal, "ChatGPT for PLC/DCS Control Logic Generation," *arXiv preprint arXiv:2305.15809*, 2023.

[27] P. Vrba, P. Tichỳ, V. Mařík, K. H. Hall, R. J. Staron, F. P. Maturana, and P. Kadera, "Rockwell Automation's Holonic and Multiagent Control Systems Compendium," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 41, no. 1, pp. 14–30, 2010.

[28] P. D. O'Brien and R. C. Nicol, "FIPA—Towards a Standard for Software Agents," *BT Technology Journal*, vol. 16, pp. 51–59, 1998.

[29] F. Bellifemine, A. Poggi, and G. Rimassa, "JADE–A FIPA-Compliant Agent Framework," *Proceedings of PAAM*, vol. 99, no. 97-108, p. 33, 1999.

[30] OpenAI, "Function Calling." [Online]. Available: https://platform. openai.com/docs/guides/function-calling

[31] HelmanCNC, "CNC Mill Program Example Outer Contour Cutting with Drilling and Counterbore," Jun 2014. [Online]. Available: https://www.helmancnc.com/

[32] L. Xia, C. Li, C. Zhang, S. Liu, and P. Zheng, "Leveraging error-assisted fine-tuning large language models for manufacturing excellence," *Robotics and Computer-Integrated Manufacturing*, vol. 88, p. 102728, 2024.

[33] J. Lim, L. Pfeiffer, F. Ocker, B. Vogel-Heuser, and I. Kovalenko, "Ontology-Based Feedback to Improve Runtime Control for Multi-Agent Manufacturing Systems," pp. 1–7, 2023.

[34] F. Joublin, A. Ceravola, P. Smirnov, F. Ocker, J. Deigmoeller, A. Belardinelli, C. Wang, S. Hasler, D. Tanneberg, and M. Gienger, "CoPAL: Corrective Planning of Robot Actions with Large Language Models," *arXiv preprint arXiv:2310.07263*, 2023.