Collective Wisdom in Language Models: Harnessing LLM-Swarm for Agile Project Management

Tahmid Hussain[1], **Tashin Ahmed**[2], **Md Shahedul Haque**[3]
Mohammad Rifat Ahmmad Rashid[4]
[1]University of Florida, [2]Cross Compass, Ltd., Japan
[3]Virginia Polytechnic Institute and State University
[4]East West University, Bangladesh
hussain.tahmid@ufl.edu, tashin.ahmed@cross-compass.com
mdshahedul@vt.edu, rifat.rashid@ewubd.edu

Abstract

The advent of large language models (LLMs) has had a profound impact on our society, providing unparalleled capabilities in a wide range of fields. However, the high expenses of developing and dealing with LLMs limit their widespread implementation. In today's fast-paced tech industry, managing complex projects efficiently remains a constant challenge. Organizations are increasingly seeking innovative technologies to optimize project management methodologies, particularly within agile frameworks. This conceptual study presents a methodology that leverages multi-agent LLMs to address these challenges, allowing organizations to effectively capitalize on the benefits of LLMs in project management. The implementation of a multi-agent LLM system can integrate diverse user perspectives by assigning distinct personalities to the agents, enhancing the system's ability to simulate context-aware interactions. The LLM-Swarm system, when utilized in the context of agile project management, offers a comprehensive understanding of projects by integrating various viewpoints through interconnected agent clusters that represent different roles, including managers, lead engineers, UI/UX designers, and quality assurance personnel. Our findings indicate that LLM-Swarm can significantly improve resource allocation, task prioritization, and overall project outcomes in agile environments.

1 Introduction

LLMs have started a new era of technological progress that is changing many parts of our culture [2, 15].

A major challenge in the development and deployment of LLMs is the substantial cost, which limits their scalability and hinders the full utilization of their potential. This study explores the integration of multi-agent intelligence with LLMs to form an LLM-Swarm system (swarmlet, in our case), building on existing concepts in both fields.

This collaborative approach utilizes the creative potential of diverse user perspectives, enabling comprehensive and multifaceted responses to complex queries. This method works especially well in project management, where people with different jobs and points of view come together.

Thorsten Händler [6] explores the limitations of LLMs and introduces autonomous LLM-powered multi-agent systems as a solution. It proposes a taxonomy to analyze how these systems balance autonomy and alignment, aiming to empower researchers in developing more reliable AI solutions.

Workshop on Open-World Agents (OWA-2024) at 38^{th} Conference on Neural Information Processing Systems (NeurIPS 2024), Vancouver, Canada.

MetaGPT [7] integrates human workflows into LLM-based multi-agent systems, employing Standardized Operating Procedures (SOPs) to streamline complex tasks and improve solution coherence, outperforming previous chat-based systems in software engineering benchmarks. [10] introduced collaborative generative agents within LLM-based systems, enhancing their coordination skills in task-oriented social contexts, exemplified in a simulated job fair, showcasing promising performance yet uncovering limitations in handling more complex coordination tasks. AlpacaFarm [5] addresses challenges in training LLMs by introducing a cost-effective simulator for learning from feedback, demonstrating high agreement with human instructions and providing reference implementations for various learning methods, with end-to-end validation showing comparable performance to models trained on human data. SELF-DEBUGGING [3] enhances LLMs code generation by teaching them to debug predicted programs through few-shot demonstrations, achieving state-of-the-art performance on diverse benchmarks, including Spider, TransCoder, and MBPP. The approach demonstrates the ability to perform rubber duck debugging, improving accuracy and sample efficiency, even on challenging tasks without unit tests.

In the field of LLM research, the application of LLM and agent-based systems is becoming an increasingly common practice. In spite of this, we have noticed that there is a lack of application of collective intelligence as a means of leveraging the knowledge that is obtained from LLM. Through the research that we are now conducting, we are working toward the goal of maximizing the effectiveness of LLMs by utilizing the potential of collective intelligence. The goal of this study is to empower the human team to plan well ahead of time for unfamiliar project scenarios they may encounter for the first time. The swarmlet framework provides them with walkthroughs of all relevant previous sources, enabling the generation of multiple plans such as plan A, B, C and so on. These alternative plans help the team anticipate where they may encounter challenges, identify their strengths and weaknesses, and understand where improvements are needed. The purpose of this work is to provide human teams with all possible scenarios before they begin a project, so that the risks are minimized during the actual execution phase. It is important to note that, throughout this paper, LLMs have been used as examples of swarm agents within the proposed swarmlet architecture. However, medium and small-sized language models can also play a highly beneficial role in such architectures, particularly in addressing specific complex challenges.

Furthermore, this framework allows for the deployment of multiple similar agents, just as in real-life projects where we require multiple developers, QA engineers, and other roles. By simulating various agent deployments, the framework can also offer insights into the manpower needed to complete the project within the required time, guiding teams in effective resource planning.

The remainder of this paper is organized as follows. In Section 2, the diverse perspectives and effectiveness of the LLM-Swarm system across various fields are discussed. Section 3 provides a detailed overview of the proposed methodology and problem formulation. The experiments and analysis are presented in Section 4. Finally, future directions are discussed, and conclusions are drawn in Section 5.

2 Implementation Aspects of LLM-Swarm

We chose to develop independent agent architectures instead of relying on pre-existing popular LLM-based single-agent frameworks like BabyAGI([1]), AutoGPT([13]), or multi-agent frameworks like CAMEL([9]), MetaGPT([7]), etc. Eventually, we present a proof of concept, demonstrating the potential of our approach rather than offering a deployment-ready system. Our LLM-Swarm system's versatility makes it well-suited for use in different fields, where the cooperative integration of different viewpoints is crucial for efficient decision-making and problem-solving.

2.1 Project Management

The LLM-Swarm system is particularly suitable for implementation in project management contexts, particularly when there is a need for collaboration among individuals with different roles in order to achieve shared objectives. The system's architecture is designed to accommodate the complex dynamics of projects. It allows for the development of questions, resolution of challenges, and decision-making procedures to use the collective intelligence of interconnected agent clusters.

2.2 R&D (Research and Development)

The LLM-Swarm system can enhance collaborative ideation in the research and development area, where creativity is crucial. Inventor agents bring novel and imaginative ideas, while feedback agents provide constructive critiques, creating an environment that is receptive to breakthroughs. The system's flexibility enables it to cater to the varied requirements of R&D initiatives.

2.3 Strategic planning for corporations

The implementation of the LLM-Swarm system in corporate strategy planning involves the collaboration of management agents, analysts, and quality assurance agents. This guarantees that strategic decisions undergo a rigorous evaluation process, embracing a wide range of viewpoints and promoting a comprehensive comprehension of the competitive environment.

2.4 Learning settings

The LLM-Swarm system has the capability to augment collaborative learning and research endeavors in educational environments. The concept facilitates dynamic discussions among student clusters, which consist of individuals from diverse experiences and viewpoints. This fosters a deeper grasp of subjects and encourages new problem-solving approaches.

2.5 Development of new products

The LLM-Swarm can optimize the collaborative process for organizations involved in product development. Innovators provide novel concepts, challenger agents carefully examine suggested functionalities, and management agents guarantee alignment with business goals. This approach guarantees a more comprehensive and cutting-edge product development lifecycle.

2.6 Development and examination of policies

The LLM-Swarm system can provide significant advantages to governments and groups engaged in policy making. Analyst agents serve as intermediaries between politicians and experts, facilitating effective communication and ensuring that policy decisions are based on accurate and thorough information. Challenger and feedback agents play a role in improving policy objectives.

3 Proposed Methodology

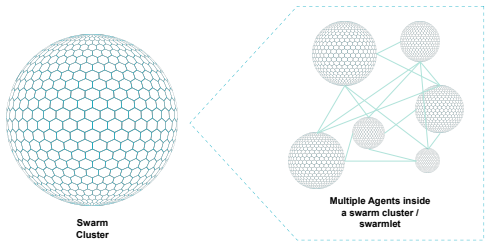

Figure 1: Illustration of a swarmlet within an cluster, representing a group of agents performing related tasks.

The proposed approach showed in Figure 1 is based on creating an LLM-Swarm system (swarmlet) that combines the knowledge of many different user groups that play important parts in project management. We refer to this as a swarmlet because it is not a full-scale swarm architecture, but rather a hand-built, small-scale demonstration. The system can be scaled up connecting more of this cluster together where the clusters represent different teams and groups designed to showcase the core principles of swarm clusters that include question generators and challenger bots, which interact dynamically to refine project scopes and challenges. These interactions are constant, ensuring that queries are evaluated from diverse perspectives (an illustration in Figure 2).

Managerial agents oversee the work of analyst agents while inventors and quality assurance agents offer a broader perspective and scrutinize the overall project. Analyst agents serve as intermediaries between managers and employees by breaking down complex information into simplified forms. Additionally, searchers bring in external information, further expanding the pool of knowledge. This

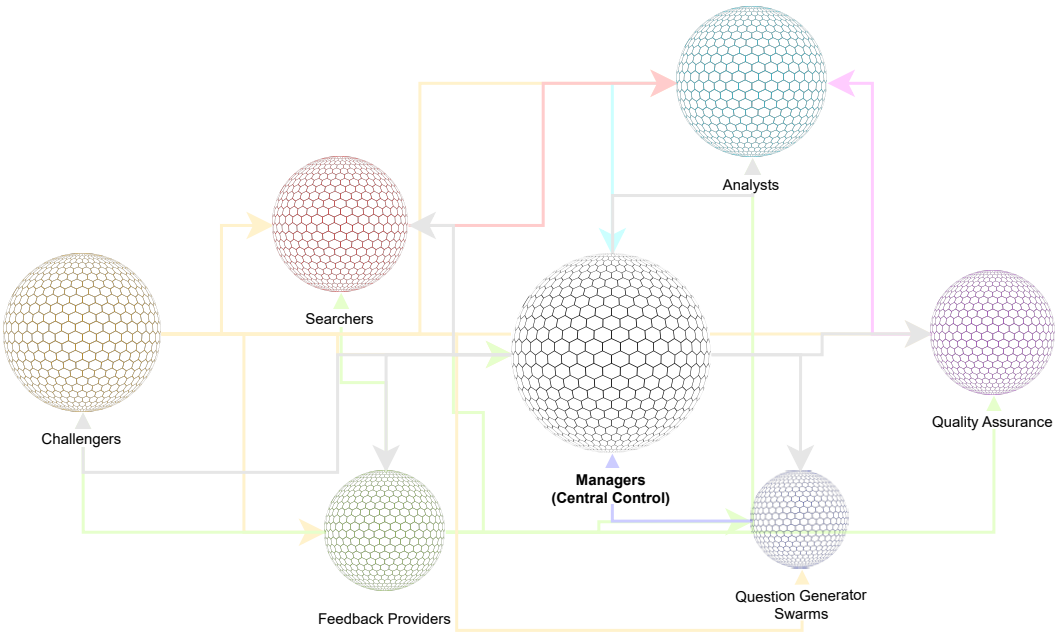

Figure 2: Proposed LLM-Swarm (swarmlet) system. **Managers** are responsible for coordinating the entire process, collaborating with all team clusters, and making final decisions. **Challengers** scrutinize every decision made by other swarm clusters in order to conduct a thorough reevaluation. The **Search** clusters perform internet searches as requested by the **Analysts** cluster. **Feedback Providers** provide feedback to all swarm clusters and strive to maintain harmony among them. The **Question Generation** cluster aids managers in formulating appropriate inquiries for other clusters. **Quality Assurance** and Innovators assist Analysts in developing a comprehensive understanding and in generating reliable and credible solutions.

collaborative swarmlet process ensures that a wide range of ideas and approaches are considered, strengthening bot the planning and execution of the project.

In this study, we have specifically applied a swarmlet to a scenario focused on project management, more precisely - agile project management. We employed agents such as the Manager, which acts as the core controller for other agents, alongside Lead Engineer, Developer, Designer, DevOps and QA. Figure 2 is a boilerplate framework that could be adapted to other scenarios such as R&D development, business operations, corporate planning and many more. However, for the purpose of this study, we demonstrate its potential in the context of agile project management in the software development field, showcasing how this architecture can enhance collaboration, decision-making, and project delivery.

3.1 Question Generation Agents

The LLM-Swarm system starts with expert agents whose job it is to come up with questions, project scopes, and problems. These agents use the different personality traits of each user to get different points of view, which sets the stage for in-depth project study.

3.2 Challenger Agents

Once the initial questions have been made, challenger agents step in and start a constructive conversation to check the truth and completeness of the questions that were made. This phase encourages a lively exchange of ideas, which helps project goals and challenges become more clear and precise.

3.3 Managerial Assessment Agents

At the same time, managerial agents look over the work that analyst agents have done to make sure that the solutions they come up with are in line with the project's goals. It is very important that these bots put together all the pieces of information and make the main plan of action based on what question generators and challengers talk about.

3.4 Analyst Agents

These agents act as go-betweens and look at the work of employee agents while also making it easier for them to talk to management agents. Their job is to turn complicated scientific information into formats that everyone can understand, make sure that everyone in different groups can work together smoothly, and help put together full project plans.

3.5 Innovators and Quality Assurance Agents

Different groups of innovators and quality assurance agents give the project a bigger picture view. Innovators bring new and creative ideas to the conversation, and quality assurance agents make sure that the suggested solutions are honest and trustworthy.

3.6 Searcher Agents

In addition to talking to each other, searcher agents look into outside sources and do searches on questions given to them by other agents. By using data from a variety of outside sources, these agents improve the general body of knowledge, which makes the process of making decisions together better.

3.7 Feedback Agents

Like challenges, feedback agents work on a larger scale and offer opinions and criticisms about the project as a whole. In addition to answering specific questions, they also give feedback on the whole process of working together and offer ways to make future interactions better.

4 Experiment and Analysis With Standalone LLM Agents

One of the ways in which the LLM-Swarm differentiates itself from a standalone LLM is by the unique approach it takes to solving problems, bringing about choices, and putting together information. The LLM-Swarm system differs from traditional isolated LLMs by integrating multiple LLM agents, promoting dynamic discourse, and facilitating collaborative decision-making. An emphasis is placed on the utilization of collective intelligence, which is achieved by the utilization of a variety of user views in order to thoroughly solve queries and adapt to emerging difficulties within a collaborative context.

4.1 Single LLM Agent with Memory

The first architecture is a single LLM agent with short-term memory designed to generate sprint plans from project specifications while maintaining context across multiple interactions. It effectively handles project management queries and produces coherent, detailed sprint plans for agile environments.

- **Core Components:** The system uses OpenAI's GPT-4 model, selected for its superior text generation compared to GPT-3.5. It is configured with a temperature of 0.2 to balance creativity and accuracy, ensuring reliable and predictable outputs suited to project management tasks.
- **Memory System:** The memory system utilizes the *ChatMessageHistory* and *RunnableWithMessageHistory* utilities from LangChain, enabling the agent to retain context across interactions. This feature is essential for ensuring consistency and referencing prior decisions in agile sprint planning.

- **Prompt Design:** A structured prompt template guides the agent's behavior by defining its role as an agile project manager and outlining its approach to sprint planning through system instructions. It includes a chat history placeholder to manage multi-step conversations and a user input section to incorporate specific project details or queries. This structure enables the system to generate sprint plans from project specifications and handle follow-up queries effectively.
- **Tool Integration:** Initially, the architecture included two tools: a Web Search Tool for retrieving up-to-date information on agile practices and software development topics relevant to sprint planning, and a Calculator Tool for performing calculations such as sprint capacity, effort estimations, and task breakdowns. However, experiments revealed that GPT-4 could perform accurate mathematical operations internally, leading to the removal of the external calculator tool in the final version of the architecture.

The input prompt for the sprint planning process is structured as follows:

$$IP_0 = SI + S + CP_0 \tag{1}$$

Here: IP_0 is the initial input prompt. SI represents the system instructions (guiding the agent's behavior), S represents the project specifications (such as user stories, backlog items), and CP_0 is the initial command prompt provided by the user.

Subsequent input prompts include previous chat history to maintain context:

$$IP_i = CP_i + CH_{i-1} \tag{2}$$

Here IP_i is the input prompt at step i, CP_i is the command prompt at step i, and CH_{i-1} is the chat history from the previous interaction.

Chat history is updated after ever interaction:

$$CH_i = CH_{i-1} + R_i + IP_i \tag{3}$$

Here, CH_i represents the newly updated chat history at step i, R_i is the response generated at step i, and IP_i is the input prompt provided for step i.

This setup ensures that the agent retains crucial information across multiple steps, allowing it to handle follow-up questions and iterative interactions more effectively.

4.2 Single ReAct Agent with Memory

The second architecture was based on the **ReAct (Reasoning + Action)** framework [17]. The key difference in this approach is that the agent does not merely generate outputs based on immediate inputs; it engages in explicit reasoning steps before taking action. This enables more thoughtful responses and better problem-solving capabilities.

Core Components: The ReAct agent also utilizes OpenAI's GPT-4 model for natural language understanding and generation. Like the single LLM agent, it employs memory through the LangChain utilities. However, it goes beyond simple question-answering by incorporating reasoning into the workflow, mimicking human-like problem-solving processes.

ReAct Framework: The ReAct agent employs a structured process of reasoning and acting, where it first reflects on the current task, evaluates its options, and decides on the most appropriate action before executing it, such as generating a sprint plan. This process includes explicit "Thought" steps, allowing the agent to assess its progress and formulate a plan. This approach ensures that the agent's decisions are deliberative rather than merely reactive, making it well-suited for complex, open-ended tasks like sprint planning.

The input prompt for the ReAct agent follows the same structure as the single LLM agent:

$$IP_0 = RP + S + CP_0 \tag{4}$$

Here, RP represents the ReAct prompt template (which adds reasoning steps), S represents the project specifications, and CP_0 is the initial command prompt.

Subsequent prompts are similar:

$$IP_i = CP_i + CHi - 1 \tag{5}$$

The chat history is updated similarly:

$$CH_i = CH_{i-1} + R_i + IP_i \tag{6}$$

This approach enables the agent to break down complex problems into manageable parts, making more accurate and informed decisions.

4.3 Multi-Agent LLM System

The third architecture developed is a multi-agent system, where multiple LLM agents with specialized roles collaborate to generate sprint plans. This approach simulates a real-world sprint planning meeting, where different stakeholders contribute from their respective areas of expertise.

Role-Based Architecture:

Each agent in the system is designated a specific role—such as Project Manager, Lead Engineer, QA Engineer, or Product Owner—and is responsible for a particular aspect of sprint planning, including task prioritization, resource allocation, testing, or quality assurance. This role-based design enables agents to concentrate on their areas of expertise while collaborating with others. For example, the Project Manager focuses on high-level goals and resource distribution, whereas the QA Engineer ensures that testing and quality control are integrated into the sprint.

Cooperative Interaction:

The agents interact and collaborate to create a unified sprint plan. For instance, the Project Manager agent may consult with the Lead Engineer agent to assess the technical feasibility and costs of specific tasks, while the QA Engineer ensures that adequate testing resources are allocated. This cooperative interaction integrates multiple perspectives, resulting in a more comprehensive and robust sprint plan compared to those generated by a single agent.

Emergent Behaviors:

A key advantage of the multi-agent system is its ability to exhibit emergent behaviors. Through collaboration, the agents may develop solutions that surpass the capability of any single agent. The interaction between agents can lead to more creative and well-rounded solutions, improving the overall quality of the sprint plans.

4.4 Evaluation Framework

To evaluate the performance of the sprint plans generated by these architectures, an LLM-based comprehensive evaluation framework was developed based on the SMART criteria [4]. The SMART framework was selected as it provides a structured evaluation approach that naturally aligns with the key aspects of sprint planning—where tasks must be specific, measurable, achievable, relevant to project goals, and time-bound within sprint durations. SMART is an established framework in the field of project management, widely used for evaluating goal-oriented tasks.

SMART Criteria:

- **Specific:** Are the tasks and goals clearly defined?
- **Measurable:** Can the outcomes be quantified or evaluated in terms of progress?
- **Achievable:** Are the proposed sprint goals realistic given the team's capacity and resources?
- **Relevant:** Do the tasks align with the overall project objectives and priorities?
- **Time-bound:** Are the tasks appropriately scoped to fit within the sprint timeline?

Table 1: Assessment for sprint plan generated by single LLM agent

Evaluator LLM	Specific	Measurable	Achievable	Relevant	Time-Bound
Meta-Llama	4	4	3	5	4
Mistral	5	5	4	5	5
GPT-4o-mini	5	4	4	5	5
Average	4.67	4.33	3.67	5	4.67

Table 2: Assessment for sprint plan generated by single ReAct(Reasoning + Action) agent

Evaluator LLM	Specific	Measurable	Achievable	Relevant	Time-Bound
Meta-Llama	4	5	3	5	4
Mistral	5	5	4	5	4
GPT-4o-mini	5	4	4	5	4
Average	4.67	4.67	3.67	5	4

We utilize a Likert scale (ranging from 1 to 5) [11, 12] to assess the quality of the output sprint plan against these five specific criteria. Each criterion is scored individually, where 1 indicates strong disagreement that the criterion is met, and 5 indicates strong agreement.Plans that score well on all aspects of the SMART framework are considered robust and effective for guiding development teams.

Our evaluation framework is inspired by a similar kind of approach introduced in [18], where human evaluators assess LLM-generated user stories based on the criteria set by the INVEST framework. We adopt a similar scoring approach but replace human evaluators with LLMs. This shift from human to LLM evaluation is driven by the need to reduce bias and enhance objectivity. For example, human evaluations can be influenced by personal connections or a lack of attention to detail, which poses a risk to inconsistencies and potential biases in the assessment. We employ three different LLMs as the judges or evaluators: GPT-4o-mini [16], Meta-Llama-3.1-8B-Instruct [14], and Mistral-7B-Instruct-v0.3 [8]. These particular models were strategically selected due to their similar parameter sizes (7-8 billion parameters), which ensures a consistent baseline for comparison. Moreover, the inclusion of both open-source and closed-source models adds diversity to the evaluation framework.

Comparison of Architectures:

- The **single LLM agent** is fast and straightforward to implement but may lack the depth that comes from considering multiple perspectives.
- The **ReAct agent** adds a layer of reasoning that helps improve the decision-making process, making it more suitable for complex projects with nuanced requirements.
- The **multi-agent system** outperforms the others in terms of generating detailed, comprehensive sprint plans but requires significantly more computational resources due to the need for multiple agents interacting simultaneously.

5 Conclusion

The integration of collective intelligence with LLMs presents an effective solution to mitigate the financial challenges of developing and utilizing LLMs. This approach can drive informed, comprehensive decision-making in critical fields like healthcare, where the stakes are high. In times of crisis, the LLM-Swarm system proves invaluable for facilitating rapid and well-grounded decisions. Within project management, LLM-Swarm offers a dynamic, collaborative framework where distinct agent clusters provide diverse perspectives. This approach not only enhances decision-making but ensures a thorough analysis of every project facet. By fostering discussions, overcoming obstacles, and integrating multiple viewpoints, the LLM-Swarm or swarmlet system has the potential to become a powerful tool for managing complex projects in an era dominated by advanced language models.

References

[1] Babyagi. `https://github.com/yoheinakajima/babyagi`. Accessed: 2024-01-19.

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[3] Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128*, 2023.

[4] George T Doran. There's a smart way to write managements's goals and objectives. *Management review*, 70(11), 1981.

[5] Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Alpacafarm: A simulation framework for methods that learn from human feedback. *arXiv preprint arXiv:2305.14387*, 2023.

[6] Thorsten Händler. Balancing autonomy and alignment: A multi-dimensional taxonomy for autonomous llm-powered multi-agent architectures. *arXiv preprint arXiv:2310.03659*, 2023.

[7] Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, et al. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*, 2023.

[8] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[9] Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in Neural Information Processing Systems*, 36:51991–52008, 2023.

[10] Yuan Li, Yixuan Zhang, and Lichao Sun. Metaagents: Simulating interactions of human behaviors for llm-based task-oriented coordination via collaborative generative agents. *arXiv preprint arXiv:2310.06500*, 2023.

[11] Rensis Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 1932.

[12] Luis M Lozano, Eduardo García-Cueto, and José Muñiz. Effect of the number of response categories on the reliability and validity of rating scales. *Methodology*, 4(2):73–79, 2008.

[13] Significant Gravitas. AutoGPT.

[14] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[15] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[16] Yiqi Wu, Xiaodan Hu, Ziming Fu, Siling Zhou, and Jiangong Li. Gpt-4o: Visual perception performance of multimodal large language models in piglet activity understanding. *arXiv preprint arXiv:2406.09781*, 2024.

[17] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023.

[18] Zheying Zhang, Maruf Rayhan, Tomas Herda, Manuel Goisauf, and Pekka Abrahamsson. Llm-based agents for automating the enhancement of user story quality: An early report. In Darja Šmite, Eduardo Guerra, Xiaofeng Wang, Michele Marchesi, and Peggy Gregory, editors, *Agile Processes in Software Engineering and Extreme Programming*, pages 117–126, Cham, 2024. Springer Nature Switzerland.