

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/379956581>

# Prioritizing Software Requirements Using Large Language Models

Preprint · April 2024

DOI: 10.13140/RG.2.2.22852.44162

CITATIONS

3

READS

338

6 authors, including:



**Malik Abdul Sami**

Tampere University

11 PUBLICATIONS 14 CITATIONS

SEE PROFILE



**Zeeshan Rasheed**

Tampere University

19 PUBLICATIONS 16 CITATIONS

SEE PROFILE



**Muhammad Waseem**

69 PUBLICATIONS 893 CITATIONS

SEE PROFILE

# Prioritizing Software Requirements Using Large Language Models

Malik Abdul Sami, Zeeshan Rasheed, Muhammad Waseem,  
Zheyang Zhang, Tomas Herda, Pekka Abrahamsson

<sup>1</sup>Tampere University, Tampere, Finland.

<sup>2\*</sup>Faculty of Information Technology, University of Jyväskylä, Jyväskylä,  
Finland.

\*Corresponding author(s). E-mail(s): [malik.sami@tuni.fi](mailto:malik.sami@tuni.fi);  
[zeeshan.rasheed@tuni.fi](mailto:zeeshan.rasheed@tuni.fi); [muhammad.m.waseem@jyu.fi](mailto:muhammad.m.waseem@jyu.fi);

Contributing authors: [zheyang.zhang@tuni.fi](mailto:zheyang.zhang@tuni.fi); [herda.tom@gmail.com](mailto:herda.tom@gmail.com);  
[pekka.abrahamsson@tuni.fi](mailto:pekka.abrahamsson@tuni.fi);

## Abstract

Large Language Models (LLMs) are revolutionizing Software Engineering (SE) by introducing innovative methods for tasks such as collecting requirements, designing software, generating code, and creating test cases, among others. This article focuses on requirements engineering, typically seen as the initial phase of software development that involves multiple system stakeholders. Despite its key role, the challenge of identifying requirements and satisfying all stakeholders within time and budget constraints remains significant. To address the challenges in requirements engineering, this study introduces a web-based software tool utilizing AI agents and prompt engineering to automate task prioritization and apply diverse prioritization techniques, aimed at enhancing project management within the agile framework. This approach seeks to transform the prioritization of agile requirements, tackling the substantial challenge of meeting stakeholder needs within set time and budget limits. Furthermore, the source code of our developed prototype is available on GitHub, allowing for further experimentation and prioritization of requirements, facilitating research and practical application.

**Keywords:** Software Engineering, Requirements Engineering, Requirements prioritization, LLMs, Generative AI

# 1 Introduction

Agile project management is a prominent methodology in software development, known for its flexibility and responsiveness to continuous changes [1]. Some leading Agile methodologies include Extreme Programming (XP), Scrum, and Kanban.[2]. Each of these methodologies has unique practices and principles designed to address specific aspects of project management and software development; for instance: fixed-length iterations and defined roles, such as the Scrum Master and Product Owner, are key components of Scrum. Kanban focuses on improving workflow through the visualization of work and the limitation of tasks. XP aims to enhance product quality by emphasizing frequent releases, simplicity, and continuous feedback [3]. In addition to these approaches, more sophisticated, global, or corporate contexts are adopting scaled agile frameworks like SAFe (Scaled Agile Framework), LeSS (Large Scale Scrum) [4, 5]. These frameworks take on the issues of scalability and ensure that Agile approaches are successfully used in bigger projects and organizations by extending Agile principles to coordinate and align different teams. However, In response to the imperatives of agile methodologies and the increasing demands for rapid development, the field of software engineering is evolving at an accelerated pace, therefore it's critical to identify effective, accurate, and adaptable methods for prioritizing Software requirements [6].

Artificial Intelligence (AI) and NLP have ushered in a new era of computational tools capable of understanding and generating text that closely mimics human writing with remarkable accuracy. The development of powerful language models, particularly those in the GPT series, marks a significant milestone in this journey [7–9]. These LLMs, with their advanced natural language understanding capabilities, stand at the forefront of a paradigm shift in software engineering. They transition from being mere tools to acting as collaborators that offer insights and analysis previously unattainable through conventional methods [10, 11]. The integration of LLMs into the requirements prioritization process is not merely an academic exploration but a necessary evolution in the dynamic landscape of software development, addressing the urgent need to automate, enhance, and refine how requirements are prioritized [12, 13].

Despite the widespread adoption of agile methodologies in software development, teams continue to struggle with effectively prioritizing project requirements. This challenge arises from the subjective nature of traditional prioritization techniques, the inherent complexity of evolving project scopes, and the difficulty of swiftly adapting to stakeholder feedback and market trends [14]. The integration of LLM-based agents in the requirements prioritization process offers a promising solution by automating the evaluation and prioritization of software requirements. However, the practical application of these technologies raises questions about their effectiveness, adaptability, and the balance between automation and human oversight in agile project management.

This study explores the potential of LLMs to improve requirements prioritization within Agile methodologies, focusing on their application and impact on software requirement engineering processes. Our investigation revolves around the following key questions:

1. How can large language models be applied in prioritizing requirements?

2. What are the limitations and opportunities of using large language models for requirements prioritization?

Our research strategy involves using LLMs to:

1. Develop a tool that captures requirements from users and generates user stories.
2. Implement prioritization logic to evaluate and prioritize user stories based on business value, urgency, and technical complexity.
3. Ensure seamless integration with project management tools like JIRA, Trello, and Azure DevOps to enhance the Agile process and stakeholder coordination.

In this paper, we present a web-based software tool for software requirement prioritization, showcasing how GPT models could redefine software engineering. Our prototype illustrates the potential of GPT agents to transform user narrative creation and requirement prioritization, empowering stakeholders and fostering collaboration. By accelerating project timelines and addressing trust, time, and budget constraints more effectively, this approach sets a new standard for stakeholder engagement and software development processes with the help of GPT models.

The paper is structured as follows: Section 2 provides a background and a survey of related literature. Section 3 describes the proposed research design and methodology. Section 4 discusses the preliminary results. Section 5 outlines limitations and future work. Finally, Section 6 concludes the paper.

## 2 Background And Related Work

### 2.1 Background: Generative AI and Its Application in Software Requirements Prioritization

Generative AI, aims to produce new material by mimicking human-like outputs in a variety of fields, including computer vision, NLP, and the creation of images and videos [15]. Text creation, machine translation, dialog systems, and code generation have all been transformed by this technology, especially through the use of autoregressive language models such as GPT and Generative Adversarial Networks (GANs) [16]. The transformer architecture, which greatly improves NLP skills by capturing contextual relationships inside text, is the foundation of the GPT model, which is well-known for its human-like text production [7, 17].

The exceptional efficiency and adaptability of GPT models, as evidenced by recent advances, indicate that they have tremendous potential to revolutionize SE processes [9, 18]. These models can streamline the software development lifecycle by automating tasks like error detection, code snippet generation, and documentation creation after being trained on large-scale code repositories [19, 20]. In addition to speeding up code generation and program development, the incorporation of GPT models into SE has improved software development's overall quality and efficiency [21, 22].

Software engineering (SE) has witnessed a rapid evolution of generative AI, demonstrating its transformational power. A major change that promises to accelerate and improve software development is the integration of GPT models into SE. These

approaches transform not just the software development process but also the way software is developed and maintained. Most importantly, they improve the prioritizing of requirements, which is an essential development stage. With the use of AI, developers can more efficiently prioritize and order work, making sure that important features match user requirements and organizational goals. The development and maintenance of software will be drastically changed by this method, which represents a shift towards more agile, user-focused, and efficient software development.

## 2.2 Related Work

An approach to needs prioritization in agile software development that is gamified is examined in the paper "PRIUS: Applying Gamification to User Stories Prioritization". To increase stakeholder participation, it blends gaming aspects with the Wieggers Matrix prioritizing technique [23]. Positive effects on engagement levels during requirements prioritizing were observed in an academic setting during the development and testing of a system that supports this technique.

Several studies (e.g., [24], [25], [12]) have explored GPT models in SE and in requirements prioritization techniques. The application of Large Language Models (LLMs) to software development, specifically for needs prioritization, is a significant advancement that holds the potential to increase both efficiency and effectiveness. However, the problems with hallucinations and the constraints of natural language identify an important research need. A solution that can not only understand broad client needs and create user stories on its own but also uses AI agents to systematically prioritize tasks is desperately needed. The latest software development approaches that emphasize agility and the creative use of LLM agents must be embraced by this instrument. By closing this gap, we hope to improve needs prioritization accuracy and dependability while also fully utilizing AI's capabilities within agile software engineering frameworks. This presents a substantial opportunity for innovative research and development in the sector.

Some established techniques and methods prioritize requirements in software development and project management. Understanding these methods matters for effective decision-making and resource allocation in enterprise-level software projects. While there are other notable techniques, we focus on those we use in our prioritization process. The Analytic Hierarchy Process (AHP) stands out. It blends mathematics and psychology, using pairwise comparisons and expert judgments to create priority scales. This process is known for its mathematical rigor and its ability to mix subjective and objective decision-making elements. Similarly, the MoSCoW method offers a straightforward yet effective framework for prioritizing requirements by sorting them into four categories: Must have, Should have, Could have, and Won't have. This method helps clarify the importance and urgency of each requirement, supporting more informed decision-making.

### 3 Preliminary Methodological Framework

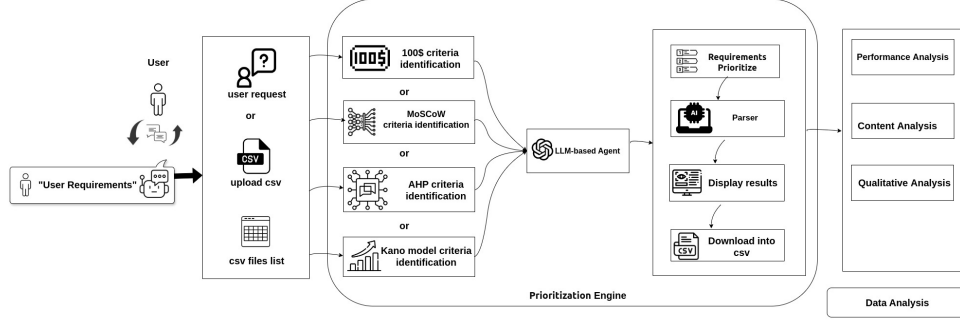
We utilize a web-based tool in our research, employing LLMs to offer a user-centric solution for the precise and efficient prioritization of software development requirements. This tool is enhanced through the integration of React, Flask, and OpenAI technologies. For a detailed illustration, we present our methodology steps and their application, providing a visual guide to our approach, the tool’s capabilities, and its output analysis.

1. **Tool Development:** The primary step involves building a web application that uses React for an intuitive front end, Flask for the back end, and OpenAI’s APIs to leverage LLMs. This application is designed to automatically generate user stories from input requirements or documents uploaded by users. It demonstrates the practical use of LLMs in creating and prioritizing software requirements according to client needs.
2. **Requirement Input and Prioritization:** Users can input their requirements or upload predefined user stories. The system uses LLMs to generate user stories and assigns them to epics randomly. Users can choose from various prioritization methods, such as AHP and MoSCoW, for a customized prioritization process, and the system displays the results.
3. **Case Study Implementation:** We conduct case studies by inputting user requirements of a Systematic Literature Review (SLR) to test the tool’s effectiveness in generating and prioritizing user stories. The tool’s capability to process and understand requirements in natural language enhances the practical evaluation, ensuring a user-friendly experience and aligning outputs closely with project needs. The prioritized list of requirements and user stories can be downloaded in CSV format for further analysis and documentation.
4. **Evaluation and Output:** During the evaluation phase, we analyze the tool’s outputs, including performance analysis, content analysis generated by LLMs, and stakeholder satisfaction through qualitative analysis.

Our methodology not only empirically validates the effectiveness of our tool but also highlights the practical advantages of integrating LLMs into the requirements engineering lifecycle. It significantly improves the efficiency and alignment of requirements prioritization with modern software development practices.

### 4 Preliminary Empirical Results

This research aims to increase the effectiveness of research procedures by introducing a system that simplifies the process of creating and ranking user stories. The system seeks to improve research workflows with the use of Python Flask API, React UI, and ChatGPT 3.5. We focus our first assessment on the creation of user stories, how they are prioritized using different methods like MoSCoW, the 100 Dollar Test, and the Analytic Hierarchy Process (AHP), as well as how the user interface is evaluated. We are ready for more investigation after this preliminary analysis. It is anticipated that a future case study, set in a real research setting, will provide hard proof of the system’s efficacy and identify areas for improvement based on user input.



**Fig. 1:** Process of Prioritizing Requirements Using LLMs and Analyzing Its Output

## 4.1 Theme and Objective

This case study presents a system engineered to automate the Systematic Literature Review (SLR) process utilizing Language Model (LLM) technologies. Its principal aim is to streamline the generation and prioritization of user stories from delineated requirements, thereby enhancing the SLR workflow's efficiency.

## 4.2 Identified User Requirements

Developed in direct response to specific user requirements, the system is designed to significantly improve research workflows. These requirements include the automated generation of research-specific questions, an intuitive user interface for seamless user-system interaction, flexibility in language model selection, efficient literature review management adhering to predefined criteria, and support for generating crucial research documentation. Meeting these requirements is essential for optimizing research output and process efficiency.

## 4.3 System Configuration

The system's architecture employs React, CSS, and JavaScript with ANT design for the front end, alongside Flask for back-end development. This configuration enables an API that interfaces with the GPT-3.5 model, facilitating the transformation of user inputs into actionable user stories through advanced natural language processing and rendering them back to the user in JSON format.

## 4.4 Step-by-Step Operation

**Step 1: Understanding user needs** The system prompts users to input their initial requirements and desired number of user stories through an engaging interface. It then processes these inputs, parses the results, and displays them promptly, as depicted in Fig 2.

**Step 2: Displaying results** The parsed data, presented in JSON format, is displayed within a dynamically generated React table (ANT design), offering a detailed view of the user stories, as shown in Figure 3.

**Fig. 2:** Requirements Gathering

No	Epic	User Stories
1	Question Generation Module	As a research scholar, I want to easily generate research-specific questions within the system to guide my investigation. -
2	React User Interface Enhancement	As a user, I want to interact with the system through a user-friendly React-based interface for a seamless experience. -
3	ChatGPT Integration	As a researcher, I want the option to choose between ChatGPT 3.5 or 4 for language modeling to suit my research needs. -
4	API Integration Framework	As a user, I need the system to support API integration for seamless data sharing and analysis. -
5	Literature Review Criteria Management	As a scholar, I aim to adhere to specific inclusion and exclusion criteria for literature review within the system. -
6	Paper Summarization and Data Extraction Module	As a user, I want the system to facilitate paper summarization and data extraction to save time and effort. -
7	Qualitative and Quantitative Analysis Tools	As a researcher, I require support for both qualitative and quantitative analysis within the system. -
8	Research Query Resolution Feature	As a user, I seek direct addressing of research queries within the system for quick and accurate results. -
9	Research Documentation Generation Tool	As a scholar, I need assistance in producing key research documentation such as abstracts, introductions, and LaTeX -

**Fig. 3:** Generation of User Stories by LLMs

**Step 3: Prioritizing Requirements** Subsequently, the system provides a feature for prioritizing these requirements using various techniques, including the Analytic Hierarchy Process (AHP). Users input their prioritization criteria, and the system organizes the user stories accordingly, converting the prioritized list back into JSON format. This prioritization process is illustrated in Figure 4.

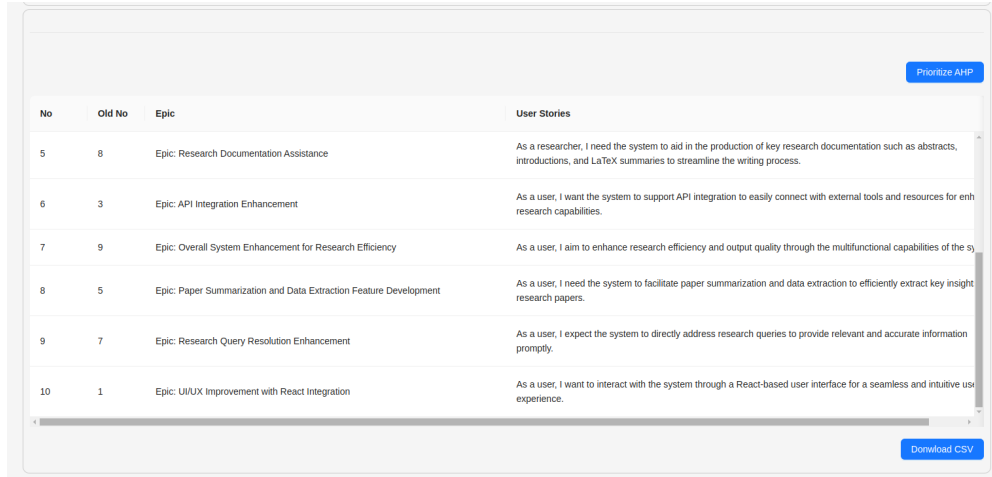
**Step 4: Exporting Requirements:** Ultimately, the system makes it easy to export prioritized requirements in CSV format. This allows for simple integration into project management tools such as JIRA, Azure DevOps, and RE management tools.

## 4.5 AHP Analysis and Case Study Performance

This section explores the system’s capacity to automate essential aspects of the SLR process effectively, particularly highlighting its proficiency in generating and prioritizing user stories based on specific user needs through the Analytic Hierarchy Process (AHP). Our approach uses AHP for user story prioritization, focusing on three main criteria: Business Value, Technical Feasibility, and User Impact, which collectively ensure that development efforts align with business objectives, maintain technical integrity, and significantly improve user experiences.

Efficiency is demonstrated by the AHP-based prioritization process completed in an average of eight seconds, showcasing the method’s effectiveness and the system’s capability to quicken the user story generation and prioritization phases. Moreover, initial observations reveal the system’s efficiency in generating epics and user stories from a minimal set of input requirements, significantly outperforming manual methodologies.





No	Old No	Epic	User Stories
5	8	Epic: Research Documentation Assistance	As a researcher, I need the system to aid in the production of key research documentation such as abstracts, introductions, and LaTeX summaries to streamline the writing process.
6	3	Epic: API Integration Enhancement	As a user, I want the system to support API integration to easily connect with external tools and resources for enhanced research capabilities.
7	9	Epic: Overall System Enhancement for Research Efficiency	As a user, I aim to enhance research efficiency and output quality through the multifunctional capabilities of the system.
8	5	Epic: Paper Summarization and Data Extraction Feature Development	As a user, I need the system to facilitate paper summarization and data extraction to efficiently extract key insights from research papers.
9	7	Epic: Research Query Resolution Enhancement	As a user, I expect the system to directly address research queries to provide relevant and accurate information promptly.
10	1	Epic: UI/UX Improvement with React Integration	As a user, I want to interact with the system through a React-based user interface for a seamless and intuitive user experience.

**Fig. 4:** Prioritization of Requirements Using LLMs

## 5 Limitations and Future work

Although this study uses Large Language Models (LLMs) to improve requirements prioritization within agile approaches, it acknowledges several limitations that open up new ways for future research. Mostly, LLM hallucinations pose a difficulty; these models tend to produce outputs that don't match the input data [26]. So we have to undertake a amount of testing and validation to make sure the generated user stories are reliable. Though limited in scale, we provide preliminary proof of concept that highlights the potential of LLMs in simplifying the requirements prioritizing process through the creation of basic user stories from requirements.

Moreover, the cost of employing services like OpenAI highlights an important factor for broad adoption, especially when it comes to projects with limited resources. A possible obstacle to the accessibility of these technologies is the need for a comprehensive assessment of the costs and benefits of powerful artificial intelligence capabilities.

### 5.1 Future work

Building upon the insights gained from this research, our future work will explore several avenues:

1. **Exploration of Open Source LLMs:** We plan to evaluate and integrate open-source LLMs to provide cost-effective alternatives and enable a comparative analysis of different models' performance. This will also enhance the system's natural language processing capabilities for improved accuracy and relevance in user story generation.
2. **Incorporation of Diverse Prioritization Techniques:** To improve versatility and effectiveness, the tool will be expanded to support additional prioritization methodologies. This adaptation will cater to a broader range of user needs and research contexts.

3. **User-Driven Story Generation:** The system will be further refined to generate user stories that precisely align with specific user requirements, ensuring relevance and actionability. Adaptive learning mechanisms will be incorporated to evolve the system based on user interactions and feedback.
4. **Co-pilot Development for Requirement Generation and Prioritization:** A co-pilot feature will be built to automate the generation and prioritization of user requirements using advanced AI techniques such as RAG and lang-chain. This feature will provide users with a guided, interactive experience in defining and organizing their research activities.

## 6 Conclusions

In this study, we presented a tool that uses OpenAI, Flask, and React to automatically generate and rank user stories based on their fundamental requirements. The application marks a major advancement in using AI to improve project management workflows by allowing users to input requirements and then generate user stories and epics for prioritization through an AI agent. The system’s ability to simplify project management activities is demonstrated by its ability to parse responses into an approachable JSON format and enable CSV downloads for connection with task management platforms like JIRA and Trello.

## References

- [1] Lei, H., Lai, W., Feaster, W., Chang, A.C.: Artificial intelligence and agile project management. In: Intelligence-Based Cardiology and Cardiac Surgery, pp. 401–405. Elsevier, ??? (2024)
- [2] Mnkandla, E., Dwolatzky, B.: A survey of agile methodologies. Transactions of the South African Institute of Electrical Engineers **95**(4), 236–247 (2004)
- [3] Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J.: Agile software development methods: Review and analysis. arXiv preprint arXiv:1709.08439 (2017)
- [4] Carroll, N., Conboy, K., Wang, X.: From transformation to normalisation: An exploratory study of a large-scale agile transformation. Journal of Information Technology **38**(3), 267–303 (2023)
- [5] Saklamaeva, V., Pavlič, L.: The potential of ai-driven assistants in scaled agile software development. Applied Sciences **14**(1), 319 (2023)
- [6] Svensson, R.B., Torkar, R.: Not all requirements prioritization criteria are equal at all times: A quantitative analysis. Journal of Systems and Software **209**, 111909 (2024)
- [7] Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al.: Improving language understanding by generative pre-training (2018)
- [8] Rasheed, Z., Waseem, M., Ahmad, A., Kemell, K.-K., Xiaofeng, W., Duc, A.N., Abrahamsson, P.: Can large language models serve as data analysts? a multi-agent assisted approach for qualitative data analysis. arXiv preprint arXiv:2402.01386 (2024)
- [9] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al.: Training language models to follow

- instructions with human feedback. *Advances in Neural Information Processing Systems* **35**, 27730–27744 (2022)
- [10] Sami, A.M., Rasheed, Z., Kemell, K.-K., Waseem, M., Kilamo, T., Saari, M., Duc, A.N., Systä, K., Abrahamsson, P.: System for systematic literature review using multiple ai agents: Concept and an empirical evaluation. *arXiv preprint arXiv:2403.08399* (2024)
  - [11] Rasheed, Z., Waseem, M., Kemell, K.-K., Xiaofeng, W., Duc, A.N., Systä, K., Abrahamsson, P.: Autonomous agents in software development: A vision paper. *arXiv preprint arXiv:2311.18440* (2023)
  - [12] Tikayat Ray, A., Cole, B.F., Pinon Fischer, O.J., Bhat, A.P., White, R.T., Mavris, D.N.: Agile methodology for the standardization of engineering requirements using large language models. *Systems* **11**(7), 352 (2023)
  - [13] Zhang, Z., Rayhan, M., Herda, T., Goisau, M., Abrahamsson, P.: Llm-based agents for automating the enhancement of user story quality: An early report. *arXiv preprint arXiv:2403.09442* (2024)
  - [14] Tomic, D.: Artificial Intelligence-driven web development and agile project management using OpenAI API and GPT technology: A detailed report on technical integration and implementation of GPT models in CMS with API and agile web development for quality user-centered AI chat service experience (2023)
  - [15] Hacker, P., Engel, A., Mauer, M.: Regulating chatgpt and other large generative ai models. In: *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*, pp. 1112–1123 (2023)
  - [16] Aydın, Ö., Karaarslan, E.: Is chatgpt leading generative ai? what is beyond expectations? *What is beyond expectations* (2023)
  - [17] Rasheed, Z., Waseem, M., Systä, K., Abrahamsson, P.: Large language model evaluation via multi ai agents: Preliminary results. In: *ICLR 2024 Workshop on Large Language Model (LLM) Agents*
  - [18] Liu, X., Zheng, Y., Du, Z., Ding, M., Qian, Y., Yang, Z., Tang, J.: Gpt understands, too. *AI Open* (2023)
  - [19] Feng, Y., Vanam, S., Cherukupally, M., Zheng, W., Qiu, M., Chen, H.: Investigating code generation performance of chat-gpt with crowdsourcing social data. In: *Proceedings of the 47th IEEE Computer Software and Applications Conference*, pp. 1–10 (2023)
  - [20] Treude, C.: Navigating complexity in software engineering: A prototype for comparing gpt-n solutions. *arXiv preprint arXiv:2301.12169* (2023)
  - [21] Dong, Y., Jiang, X., Jin, Z., Li, G.: Self-collaboration code generation via chatgpt. *arXiv preprint arXiv:2304.07590* (2023)
  - [22] Ma, W., Liu, S., Wang, W., Hu, Q., Liu, Y., Zhang, C., Nie, L., Liu, Y.: The scope of chatgpt in software engineering: A thorough investigation. *arXiv preprint arXiv:2305.12138* (2023)
  - [23] Lencastre, M., Silva, D., Pimentel, J.H.C., Castro, J.B.: Prius: Applying gamification to user stories prioritization. *ACM SIGAPP Applied Computing Review* **23**(4), 27–44 (2024)

- [24] Ahmad, A., Waseem, M., Liang, P., Fahmideh, M., Aktar, M.S., Mikkonen, T.: Towards human-bot collaborative software architecting with chatgpt. In: Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering, pp. 279–285 (2023)
- [25] Barke, S., James, M.B., Polikarpova, N.: Grounded copilot: How programmers interact with code-generating models. Proceedings of the ACM on Programming Languages **7**(OOPSLA1), 85–111 (2023)
- [26] Xu, Z., Jain, S., Kankanhalli, M.: Hallucination is inevitable: An innate limitation of large language models. arXiv preprint arXiv:2401.11817 (2024)