future internet

Review

Using ChatGPT in Software Requirements Engineering: A Comprehensive Review

Nuno Marques [1], Rodrigo Rocha Silva [2,3] and Jorge Bernardino [1,2,*]

[1] Coimbra Institute of Engineering—ISEC, Polytechnic University of Coimbra, Rua Pedro Nunes, 3030-199 Coimbra, Portugal; a21230094@isec.pt
[2] Centre for Informatics and Systems of the University of Coimbra (CISUC), Pólo II, Pinhal de Marrocos, 3030-290 Coimbra, Portugal; rrochas@dei.uc.pt
[3] FATEC Mogi das Cruzes, São Paulo Technological College, Mogi das Cruzes 08773-600, Brazil
* Correspondence: jorge@isec.pt

Abstract: Large language models (LLMs) have had a significant impact on several domains, including software engineering. However, a comprehensive understanding of LLMs' use, impact, and potential limitations in software engineering is still emerging and remains in its early stages. This paper analyzes the role of large language models (LLMs), such as ChatGPT-3.5, in software requirements engineering, a critical area in software engineering experiencing rapid advances due to artificial intelligence (AI). By analyzing several studies, we systematically evaluate the integration of ChatGPT into software requirements engineering, focusing on its benefits, challenges, and ethical considerations. This evaluation is based on a comparative analysis that highlights ChatGPT's efficiency in eliciting requirements, accuracy in capturing user needs, potential to improve communication among stakeholders, and impact on the responsibilities of requirements engineers. The selected studies were analyzed for their insights into the effectiveness of ChatGPT, the importance of human feedback, prompt engineering techniques, technological limitations, and future research directions in using LLMs in software requirements engineering. This comprehensive analysis aims to provide a differentiated perspective on how ChatGPT can reshape software requirements engineering practices and provides strategic recommendations for leveraging ChatGPT to effectively improve the software requirements engineering process.

Keywords: ChatGPT; LLMs; software engineering; software requirements; generative AI

check for updates

Citation: Marques, N.; Silva, R.R.; Bernardino, J. Using ChatGPT in Software Requirements Engineering: A Comprehensive Review. *Future Internet* **2024**, *16*, 180. https://doi.org/10.3390/fi16060180

Academic Editor: Gianluigi Ferrari

Received: 15 March 2024
Revised: 10 May 2024
Accepted: 16 May 2024
Published: 21 May 2024

1. Introduction

Software requirements engineering is a critical phase in the software development lifecycle that lays the foundation for successfully delivering software products that meet stakeholders' needs and expectations. Traditionally, this process involves eliciting, analyzing, specifying, and validating requirements, often time-consuming, labor-intensive, and prone to human error and bias. Herein lies the potential for large language models (LLMs) to increase and streamline these processes by providing efficient and effective solutions for requirement elicitation, documentation, and validation [1]. An IEEE consolidated definition for a software requirement is a documented condition or capability that a system or system component meets or possesses to solve a contractual, standard, specification problem, or objective [2].

In recent years, advances in natural language processing (NLP) have paved the way for transformative applications in diverse domains, such as healthcare, finance, e-commerce, social media, and more. Among these breakthroughs, LLMs have emerged as powerful tools with the potential to revolutionize software engineering practices, particularly in software requirements engineering.

LLMs like ChatGPT have unprecedented capabilities for understanding, generating, and processing natural language text. Their vast pre-trained knowledge and ability to generate contextually relevant text make them promising candidates to assist and automate various tasks in software development [3]. However, integrating LLMs into software requirements engineering presents challenges and concerns.

While LLMs exhibit impressive capabilities in generating coherent and contextually relevant text, they may also produce inaccuracies, ambiguities, or biased outputs, posing risks to the reliability and quality of software requirements. Moreover, ethical considerations such as data privacy, model bias, and transparency justify careful examination when employing LLMs in sensitive domains like software development [4]. A critical aspect of using LLMs effectively is formulating appropriate prompts—the input text or queries provided to these models to generate desired outputs [5].

Prompt patterns, defined as structured and systematic approaches to creating prompts for LLMs, have emerged as critical tools for realizing the full potential of these models. Researchers and practitioners can guide LLMs to produce outputs that align more closely with their intended objectives by defining specific patterns or templates for prompts. These patterns facilitate the generation of coherent and contextually relevant text and enable finer control over the outputs generated by LLMs [6,7].

This study aims to evaluate the impact of ChatGPT on the software requirements elicitation process. To achieve this goal, we investigated the benefits and challenges of this Generative AI (GenAI) based on studies conducted by other authors and provided a critical analysis of these studies.

The main contributions of this work are the following:

- Understanding the role of GenAI in software requirements;
- Characterizing the benefits and challenges of using ChatGPT to assist with software requirements;
- Identifying future research directions.

Considering the transformative potential of large language models (LLMs) like ChatGPT in software requirements engineering, this research seeks to answer the following research question:

RQ: How can ChatGPT be used to improve software requirements engineering processes, and what are the associated benefits, challenges, and ethical considerations?

To address this question, we conducted a literature review evaluating the application of ChatGPT in software requirements engineering. Through this analysis, we elucidate the role of ChatGPT in streamlining requirement elicitation, documentation, and validation while characterizing the benefits and delineating the challenges of integrating ChatGPT in this domain. Additionally, we explore the ethical considerations and propose future research directions to optimize the use of LLMs in software requirements engineering, providing a foundation for further exploration and development in this area.

The rest of this paper is structured as follows: Section 2 provides some background about software requirements engineering, LLMs, and an overview of ChatGPT. Section 3 describes the methodology used to conduct the literature review. Section 4 presents the literature review. Sections 5 and 6 approach the benefits and challenges of using ChatGPT in software requirements engineering, respectively. Section 7 outlines future research directions. Finally, Section 8 presents the conclusions.

2. Background

While the application of artificial intelligence/machine learning (AI/ML) in software engineering research has a long history, the specific use of GenAI is a more recent and emerging topic. Although the potential of GenAI has been recognized for some time, research progress in this area has been rapid, particularly since 2020. Despite some earlier exploration of GPT-2 for code generation, GenAI has recently gained significant attention in software engineering. Recent advances in these systems, particularly introducing services such as GitHub Copilot and ChatGPT-4, have spurred increased research interest in various disciplines, including software engineering [8]. In this section, we provide an overview of the basic concepts of software requirements engineering, LLMs, and an overview of ChatGPT.

2.1. Software Requirements Engineering

Software requirements engineering is a systematic and process-driven approach to defining, documenting, and maintaining software requirements across the software development lifecycle [1]. This multidimensional task requires robust information retrieval, effective communication with multiple stakeholders, and the creation of detailed textual descriptions. It comprises two main phases: requirements development and management [9]:

- **Requirements development:** This involves activities such as eliciting, analyzing, specifying, and validating the requirements.

The requirements elicitation phase involves identifying stakeholders, selecting representatives, and determining their needs. It serves as the information-gathering step in requirements development. Various techniques for requirements elicitation include stakeholder interviews, focus groups, workshops, observations, questionnaires, document analysis, and benchmarking.

The requirements analysis step synthesizes and refines information gathered during requirements elicitation. Stakeholder needs, assumptions, and other information are integrated and further detailed. This phase involves representing requirements in various forms, such as prototypes and models, conducting trade-off analysis, setting priorities, assessing feasibility, and identifying gaps to uncover missing requirements.

A recommended practice for requirements specification is to utilize predefined templates. These templates enable requirements engineers to concentrate on content rather than format, reducing the risk of overlooking critical items while documenting requirements.

The final step in the requirements development process is to validate the requirements to ensure they are well-written, complete, and aligned with customer needs. Validation may result in iterations of previous steps due to identified defects, gaps, additional information or analysis requirements, clarification, or other issues.

- **Requirements management:** This includes the processes for requesting changes to established requirements, performing impact analysis on those changes, approving or rejecting them, and then implementing the approved changes.

Requirements management is a continuous process that spans the entire software development lifecycle. During testing, the software product is validated against its requirements to identify and correct defects, ensure that it meets the specified requirements, and provide confidence in its functionality [9].

2.2. Large Language Model (LLM)

A large language model (LLM) is a deep learning algorithm that can perform a variety of tasks in natural language processing (NLP). Large language models use transformer models and are trained using massive datasets. This enables them to recognize, translate, predict, or generate text or other content [4]. LLMs such as InstructGPT and GPT-4 excel at in-context learning and generating coherent and contextually relevant responses based on given prompts. Reinforcement Learning from Human Feedback (RLHF) is a fundamental technique for LLMs. It consists of incrementally improving the model's performance by using human-generated responses as feedback [4].

Large language models (LLMs) are gaining popularity in academia and industry due to their unprecedented performance in various applications. As LLMs continue to play an important role in research and daily use, their evaluation becomes increasingly important, not only at the task level but also at the societal level, to better understand their potential risks. In recent years, significant efforts have been made to study LLMs from different perspectives.

A common approach to interacting with large language models (LLMs) is prompt engineering, where users design specific prompts to guide LLMs to generate desired responses or complete tasks. This approach is widely used in evaluation efforts. Furthermore, users can interact with LLMs in question-and-answer or dialogue mode, encouraging natural language conversations.

In summary, LLMs, leveraging transformer architecture, in-context learning, and Reinforcement Learning from Human Feedback (RLHF), have revolutionized natural language processing (NLP) and show promise in several applications, such as ChatGPT and GitHub Copilot [6].

2.3. Overview of ChatGPT

ChatGPT is an AI chatbot built using OpenAI's large language models (LLMs), such as GPT-4 and earlier versions [10]. It has set new standards (tasks, metrics, etc.) in artificial intelligence by demonstrating machines' ability to understand the intricacies of human language and communication.

OpenAI unveiled an initial demonstration of ChatGPT on 30 November 2022, sparking widespread interest on social media as users demonstrated its capabilities. Examples ranged from travel planning to creating fables to coding computer programs. Within five days, the chatbot had more than one million users. ChatGPT's evolution has been one of continuous improvement, with each version building on the foundation laid by its predecessors.

This evolution is also characterized by an exponential increase in the number of parameters used to train the model. This increase in parameters generally leads to significant improvements in the effectiveness of the solutions presented by the models. This is because models with more parameters have a greater ability to learn and capture complex nuances and patterns in the training data.

Chat GPT-4, the latest version, continues this trend of exponential improvement with changes such as improved model alignment, Internet connectivity, better steerability, and more [10].

The GPT model and software engineering intersect by applying natural language processing (NLP) techniques to various tasks within the software development lifecycle. GPT's language generation capabilities provide valuable support and enhancements to software engineering processes [11].

ChatGPT can streamline the process of gathering, analyzing, and documenting requirements in the context of software requirements. It uses its extensive knowledge base and language understanding capabilities to engage stakeholders in conversations, generate accurate documentation, and provide real-time feedback on requirements. Through its ability to continuously learn and adapt, ChatGPT improves stakeholder collaboration, accelerates the requirements gathering process, and ultimately contributes to more efficient and successful software development outcomes. However, ChatGPT should not be viewed as a replacement for human expertise and judgment. Instead, ChatGPT should be viewed as a complement to it [12].

3. Methodology

This section outlines the approach taken in conducting the literature review, which involved synthesizing existing knowledge, critically assessing methodologies, and analyzing results to compare the use of ChatGPT to improve the quality of software requirements. The process is illustrated in Figure 1 and explained in the following sections.

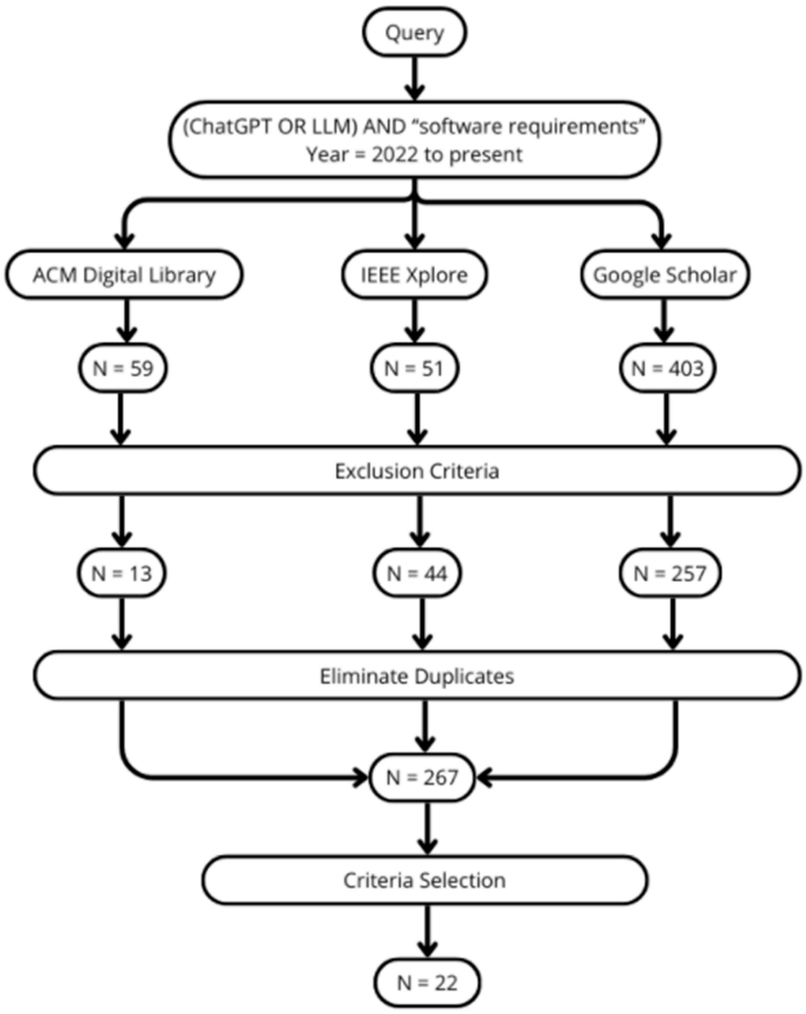

Figure 1. The methodology used in the literature research.

3.1. Data Sources

Google Scholar (www.googlescholar.com) has developed over the years and has become a robust database for scientific literature [13]. Therefore, it was chosen as the main research tool for the present study. However, we add two other important public data sources: IEEE Xplore (https://ieeexplore.ieee.org/) and the ACM Digital Library (https://dl.acm.org/).

3.2. Search Queries

A search in Google Scholar on 19 April 2024, with the query "(ChatGPT OR LLM) AND ("software requirements")" returned about 403 results in just 0.04 s. The same query was used in IEEE Xplore and returned 51 results, and using the ACM Digital Library, we obtained 59 results (see Figure 1).

3.3. Inclusion Criteria

We considered the year of publication as an inclusion criteria. Because ChatGPT was launched that year, only research published from 2022 to the present was considered.

This review includes the most relevant papers from each year. Relevance in Google Scholar refers to the degree to which the search results match the criteria or context of the query. The sorting algorithm considers several factors to determine the order of the results, including the presence of search terms and citation counts.

3.4. Exclusion Criteria

The exclusion criteria were books, internal reports, theses/dissertations, citations, presentations, abstracts, and appendices. Papers written in languages other than English were also excluded.

3.5. Characterization of Selected Papers

Figure 1 shows a distribution of the total number of papers retrieved from the ACM Digital Library (59), IEEE Xplore (51), and Google Scholar (403) as data sources. The earliest paper dates from 2022, and thus, potentially, 513 papers could be selected.

Applying the exclusion criteria defined in Section 3.4, the total number of papers was reduced to 314, distributed as follows for each data source: ACM Digital Library (13), IEEE Xplore (44), and Google Scholar (257).

After removing the papers found in two or more data sources, the final result is 267. A distribution of these papers retrieved per year is shown in Figure 2. As can be seen, the rise of GenAI, exemplified by the use of models such as ChatGPT in software requirements, has catalyzed an exponential increase in research.

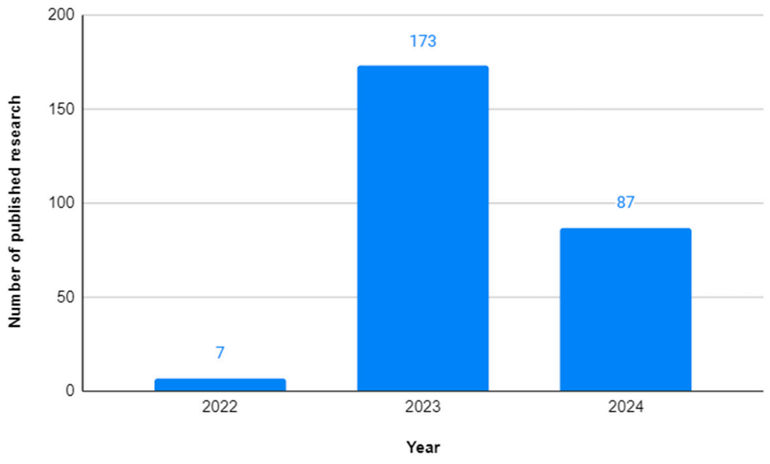

Figure 2. Evolution of research about AI impact on software requirements.

From this deduplicated number (267), we selected 22 papers for our analysis. This selection was based on each paper's title, abstract, introduction, and conclusions, with the criteria being those that use the capabilities of LLM, specifically ChatGPT, to improve the quality of software requirements.

4. Literature Review

This literature review included the most recent scientific literature on the impact of LLMs in software requirements. As mentioned above, we selected 22 papers, which are distributed by year as follows: 2022: 1, 2023: 16, and 2024: 5. In this section, the papers are presented by year, and within each year, they are presented in alphabetical order of author(s).

Liu et al. [2] emphasize the importance of requirements engineering in the software development lifecycle and discuss the role of artificial intelligence (AI) in improving requirements quality. They review recent research on how AI techniques like machine learning (ML), classification, and natural language processing (NLP) have advanced requirements engineering. The authors suggest employing appropriate ML and NLP techniques to meet textual requirements, which can extract deeper meaning and improve requirements engineering performance. The authors highlight the ability of AI to assist requirements engineers by automating mundane tasks in the requirements engineering process, allowing them to focus more on creative aspects. A strength of this study is the identification of the knowledge gap on how AI should be incorporated into software requirements processes to ensure high quality requirements representation. Although the text summarizes each recent study and its impact on the field of requirements engineering, it does not provide specific details about these results. It would be useful to include a more detailed analysis of recent studies, highlighting their key findings and how they contribute to advancing the field of requirements engineering.

Abdelfattah et al. [14] present a method for teaching software engineering using ChatGPT. This method involves using ChatGPT to create software architecture diagrams, which can be very useful in generating software requirements. The authors highlight the simplicity of understanding the basic concepts of requirements engineering through this approach. In parallel, the study highlights the importance of ChatGPT in capturing software requirements and demonstrates its application in real cases. Overall, this research confirms the effectiveness of ChatGPT in understanding and eliciting software requirements, thus consolidating the growing relevance of this tool in the field of software engineering. A strong point of the study is the emphasis on understanding requirements engineering. This emphasis is supported by the creation of practical exercises that focus on understanding the basic principles of requirements engineering, a crucial aspect of software development. The step-by-step approach to user interaction with ChatGPT in creating diagrams also facilitates understanding of the process, making it another strong research point. However, the lack of examples of requirements generated as a result of the experiments may raise doubts about the effectiveness of this method. Including concrete examples of software requirements generated by ChatGPT during the experiments could provide a more tangible validation of the usefulness and relevance of the proposed method.

Arora et al. [15] propose a SWOT analysis for research and development using LLMs for requirements engineering, focusing on the potential for requirements elicitation, analysis, specification, and validation. The authors evaluate ChatGPT's performance in classifying requirements. By analyzing the results obtained, they provide a comprehensive understanding of the effectiveness of this approach and add credibility to their conclusions. However, they suggest a cautiously optimistic outlook on the role of AI in elicitation and validation processes. Balancing the benefits of LLMs with the need for rigorous validation, ethical compliance, and human oversight is suggested to realize their full potential in requirements engineering.

Belzner et al. [16] discuss the potential benefits and challenges of using large language models (LLMs) in software engineering. They discuss the opportunities in requirements engineering, system design, code generation, test generation, code reviews, and software processes. It also thoroughly reviews current state-of-the-art methods for using LLMs in software engineering. However, a significant gap is the lack of explanation of how ChatGPT makes its decisions to achieve the desired results. This lack of transparency in decision-making can undermine developers' confidence in ChatGPT. Without understanding how the model arrives at its answers, developers are reluctant to adopt or fully trust ChatGPT's recommendations.

Bencheik et al. [1] highlight the effectiveness of ChatGPT in generating software requirements and the role of human input. Moreover, the study acknowledged the time efficiency of ChatGPT but noted that experienced human participants tend to produce more thorough requirements. They highlight the importance of using AI tools to complement human expertise in requirements engineering. Although the output generated by ChatGPT is efficient, the user still needs to verify its integrity. The authors emphasized ChatGPT's capability to mimic human expertise while underscoring the critical role of human feedback in improving requirements quality. However, the lack of detail about the practical challenges in implementing GenAI tools can impact the quality of the requirements generated. A deeper understanding of the obstacles faced in the practical application of GenAI is crucial to ensure that the proposed solutions are effective and appropriate to the needs of software development.

Cheng et al. [17] provided a study focused on integrating AI assistant prototypes into established programming environments, emphasizing systematizing methodologies and proposing design principles for AI-assisted programming. The results indicate that the user interface of the AI assistant within the integrated development environment (IDE) significantly impacts the tool's effectiveness, underscoring the importance of careful design considerations. The authors introduce Prompt Sapper, a block-style visual programming tool that enables AI chain engineers, including non-technical users, to compose prompt-based AI services using foundation models through chat-based requirements analysis and visual programming. Sapper includes two views, with LLM co-pilots to help elicit requirements, generate code skeleton, and run/test the AI service. This tool allows AI chain engineers to create prompt-based services on top of foundation models, using chat-based requirements analysis and visual programming. This makes it easier for users to understand how to use these tools. However, a weakness of this study is the lack of explanation of the principles and practices incorporated into this methodology. A more detailed description of the principles and practices underlying the Prompt Sapper tool would be beneficial to readers.

El-Hajjami et al. [18] empirically evaluated two ChatGPT models (GPT-3.5-turbo and GPT-4) for requirements classification and compared them with traditional classification methods such as support vector machines (SVM) and long short-term memory (LSTM). They concluded that there is no single best technique for all requirements classifications because the best technique varies depending on the specific requirement classification. By analyzing the results obtained, they provide a comprehensive understanding of the effectiveness of this approach and add credibility to their conclusions. However, a significant gap is the lack of explanation of how ChatGPT makes its decisions to achieve the desired results. This lack of transparency in decision-making can undermine developers' confidence in ChatGPT. Developers are reluctant to adopt or fully trust ChatGPT's recommendations without understanding how the model arrives at its answers.

Kutzner et al. [19] conducted a study that provided insights into the potential and challenges of generating texts for requirements specification in software development using AI. The authors highlighted the need for humans to control the process, regardless of AI's accuracy and effectiveness in software tasks. Also, implementing AI text generation to create requirements and functional specifications has facilitated documentation creation. The study's strength is recognizing the crucial role of requirements and functional specifications in the software development process. These specifications are key to facilitating communication and promoting effective understanding between stakeholders and software developers. The focus of research on improving the efficiency and accuracy of the transition from functional and non-functional requirements to functional specifications further raises the importance of these aspects. However, the failure to validate the results obtained through testing in real software development projects to assess their practical usefulness and effectiveness can be considered a limitation of the study. This practical validation is crucial to understanding the complexities and challenges encountered in professional contexts.

Qian et al. [20] propose ChatDev, a chat-based software development framework that leverages large language models (LLMs) throughout the entire software development process, streamlining and unifying key processes through natural language communication, thereby eliminating the need for specialized models at each phase. The research highlights the improvement of the desired output by decomposing the software development process into sequential subtasks. The results are optimistic, praising the efficiency and cost-effectiveness of the software development process using ChatDev. The most significant implication is the approach and exploration of the use of large-scale language models (LLMs) throughout the software development process, thus eliminating the need for specialized models for each phase. This approach promises to reduce complexity and make communication between developers more efficient, contributing to a more fluid and cohesive integration of the various stages of software development. However, a weak point of this study is the lack of empirical experiments to support the results obtained. The lack of concrete evidence from empirical experience may reduce the credibility and confidence of the conclusions presented in the study.

Ronanki et al. [3] investigated the potential of ChatGPT to support software requirements. Six requirements elicitation questions were formulated, and interview-based surveys were conducted with requirements engineering experts. The results showed that ChatGPT-generated requirements scored higher than human-generated requirements on several quality attributes. They highlight the need for further research to effectively use LLMs in natural language-based requirements engineering activities. One of the strengths of this study is the active participation of requirements engineering experts, which contributes to increasing the reliability of the results. However, it is important to note that this evaluation is based exclusively on the opinions of experts, which can introduce bias into the analysis. Incorporating additional validation techniques could increase the effectiveness of the evaluation of the requirements generated.

Rasheed et al. [21] explored how multiple GPT agents can perform various software engineering tasks, requirements analysis, design, code generation, debugging, and managing software maintenance autonomously. The results show a reduction in development time and advanced code generation methods, reinforcing the potential of AI-driven software engineering practices to make software development more efficient, accessible, and innovative. The strength of this study is the experimental analysis that demonstrates the capabilities of ChatGPT during the various phases of software requirements. Another strong point is the creation of a roadmap for future work, highlighting the intention to improve ChatGPT's capabilities. However, it is important to emphasize that the study is based on an experimental analysis performed on simple software projects. This may limit the generalization of the results to more complex projects in software engineering practice, which is a weakness of this study.

Sridhara et al. [12] used ChatGPT to investigate everyday software engineering tasks such as resolving ambiguity in software requirements, suggesting method names, prioritizing test cases, reviewing code, and summarizing logs. They compared responses generated by ChatGPT with corresponding outputs from human experts. They conclude that ChatGPT does very well for specific tasks but not so well for others that it cannot provide answers. They concluded that ChatGPT can be employed independently or as an auxiliary tool for software engineering tasks.

The study conducted by Subahi [22] presents a mechanism for mapping software requirements using the BERT language model in conjunction with a large dataset. Four main criteria were considered to evaluate the model's effectiveness: precision, accuracy, recall, and F1 score. After a series of tests, the model proved to be a significant advance in understanding software requirements, achieving high performance in all the criteria evaluated. However, despite the model's promising potential, the author highlights the need for adjustments depending on the importance attributed to each criterion. This adjustment aims to balance precision and recall, ensuring a more complete and practical approach to software requirements analysis. A significant shortcoming of this work is the lack of consideration of functional requirements in the dataset used in the experiment. Functional requirements play a crucial role in software development, describing the specific functionalities that the system must provide to meet users' needs. By not including functional requirements in the experiment dataset, the analysis may be incomplete and not fully reflect the complexity and real challenges faced in the requirements engineering process.

Wang et al. [23] introduced a framework called Chatcoder to refine the requirements by chatting with LLMs. They design a chat scheme in which the LLMs will guide the human users to refine their expression of requirements to be more precise, unambiguous, and complete than before. It then presents the refined arguments back to the users in an understandable way. This represents a significant advance, as it provides requirements engineering professionals with a practical tool for improving the quality of software requirements. Another important contribution is the recognition of the importance of human supervision and intervention in ensuring the quality and relevance of requirements. The involvement of human users in reviewing and editing the requirements generated by LLM is fundamental to ensuring that they meet the needs and expectations of the stakeholders.

However, a weakness of this study is the limited experimental scope. Although the experiment showed positive results regarding the effectiveness of LLM in refining requirements, the scope of the experiments may be limited to specific domains or data sets. This may affect the generalizability of the results to broader contexts or real-world applications, requiring additional validation and testing in different contexts to ensure the robustness and applicability of the results.

White et al. [6,7] provided a catalog of prompt engineering design patterns for various software engineering tasks, including requirements elicitation. A list of requirements elicitation patterns is presented to aid in creating software requirements and exploring their completeness with respect to desired system capabilities and accuracy: requirements simulator, specification disambiguation, and change request simulation. All these patterns were tested with the ChatGPT. The systematic classification provides quick access to solutions for specific challenges, thereby increasing efficiency and productivity. However, this work lacks an in-depth exploration of practical implementations of the prompt patterns to thoroughly assess their effectiveness.

Zhang et al. [24] developed an approach to empirically evaluate ChatGPT's ability to retrieve information about requirements. This framework performed a preliminary evaluation of ChatGPT's zero-shot requirements retrieval performance on two requirements analysis tasks over four datasets. The evaluation results show that ChatGPT can retrieve the relevant information about requirements (high recall). However, its ability to retrieve more specific requirements information is limited (low precision). One of the most significant implications of the study is that their experiments provide valuable information on the effectiveness of various prompting strategies in interacting with ChatGPT. However, a weakness of this study is the lack of discussion about the benefits and challenges of using ChatGPT in software requirements gathering. This lack of analysis can lead developers to over-rely on ChatGPT, which may not be advisable. Without a comprehensive understanding of the benefits and challenges associated with using ChatGPT in the requirements elicitation process, developers may not be fully prepared to deal with its limitations and nuances.

Fantechi et al. [25] analyze the potential usefulness of extended language models (LLMs) for detecting requirements variability in natural language (NL) requirements documents using GPT-3.5 and Microsoft Bing. To investigate this issue, the authors carried out a preliminary experiment. They used two NL requirements documents as examples. They compared the variability detection capabilities of LLM-based chatbots with those of human experts and a rule-based natural language processing (NLP) tool. The comparative study carried out by the authors is a strength because the results obtained provide valuable information on the performance of the different approaches to variability detection. This comparison allows readers to gain insight into each approach's advantages and limitations, helping them choose the best tool according to their needs. The selection of natural language (NL) requirements documents can be considered a weakness identified in this study. The effectiveness of LLM-based tools in detecting variability can vary significantly depending on the characteristics and complexity of the NL requirements documents selected. A less comprehensive or representative selection of these documents may raise questions about the generalizability and reliability of the results obtained.

Luitel et al. [26] focused on requirements completeness and used BERT's masked language model (MLM) to generate predictions for filling masked slots in requirements. The authors used 40 requirements specifications from the PURE dataset to determine the model's accuracy. The predictions are characterized according to the following criteria: performance, efficiency, stability, accuracy, and reliability. Despite the positive results presented, the authors recognize that the lack of collaboration between software engineering experts can incorrectly identify cases of incompleteness in the requirements and consequently lead to false effectiveness. Detecting incomplete requirements is a major challenge in the field of requirements engineering, as incomplete requirements can lead to significant errors during the software development process. A strong point of the research is the emphasis on this crucial challenge. The relevance of this study is backed up by an empirical evaluation that references real requirements specifications from the PURE dataset. The results shown demonstrate the effectiveness of the proposed approach in dealing with the detection of incomplete requirements. However, it is important to note that the experiment was conducted using only half of the PURE dataset, which may limit the generalizability of the results to the full dataset.

Oswal et al. [27] highlight the importance of agile methodologies in software development, especially the concept of user stories. They emphasize that the manual generation of these stories from unstructured requirements is laborious and challenging. To solve this problem, an AI-based approach is proposed, using the GPT-3.5 language model to transform requirements texts into structured user stories efficiently. In their conclusions, the authors highlight the reduction in time and effort required to obtain user stories while also minimizing the risk of human error. They also emphasize the opportunity for professionals to focus on more valuable tasks. A strong point is the use of a real-world scenario, which effectively demonstrates how agile approaches can successfully produce software solutions. This approach gives readers a deeper understanding of the process and how it can be applied in practice. However, one weakness is the lack of validation of the results through user testing or case studies.

Waseem et al. [28] presented a study investigating the effectiveness of ChatGPT as a software development bot in different phases of the software development lifecycle, particularly projects led by students. The results highlighted skill deficits among students and a remarkable enthusiasm for AI. ChatGPT demonstrated its value as a supportive tool, fostering efficiency and collaboration. The impact demonstrated by this study on the use of ChatGPT in the academic environment is evident. However, one gap observed is the lack of an in-depth discussion of the specific advantages and limitations of using ChatGPT in the various phases of software requirements. Each phase of the software development process presents unique challenges and requirements, and understanding how ChatGPT can be most effective in each of these phases is crucial to its adoption and practical application.

Yeow et al. [29] explore the potential of ChatGPT-3.5 for automating software requirements engineering tasks by performing an analysis and evaluation of the questions generated by ChatGPT-3.5 for eliciting software requirements. The evaluation criteria include the questions' legibility, clarity, relevance, and completeness. The authors also identify the challenges and limitations of using ChatGPT-3.5 for software requirements gathering. The results of the experiments indicate a performance that meets university educational standards regarding legibility. Although the generated questions show above-average results in other criteria, the authors point out that there is still room for improvement in the approach and structure of this method. We have identified two strengths. Firstly, the authors offer a comprehensive analysis of the capabilities, limitations, and potential applications of ChatGPT-3.5 in requirements engineering, allowing the reader to gain a more effective understanding of these aspects. Secondly, they present concrete recommendations for future research focused on integrating ChatGPT-3.5 into software requirements engineering. However, a weakness of this study is the lack of an in-depth analysis of how the prompts used in the experiments were constructed. The quality of the prompt directly influences the effectiveness of the output generated by ChatGPT-3.5. Therefore, a more detailed analysis of the prompts used could provide additional insights into how to optimize the performance of ChatGPT-3.5 in generating software requirements.

We concluded that only a few papers describe in detail the benefits of ChatGPT in software requirements. Therefore, we aim to fill this gap and further explore these benefits.

Table 1 presents a comparison of the analyzed works, listing the fundamental aspects that we consider for a comprehensive analysis of scientific works that aim to discuss the use of ChatGPT for requirements generation. We attempt to provide a nuanced overview of the current research landscape by examining key features such as the effectiveness of ChatGPT in generating requirements, the indispensable role of human feedback, the exploration of prompt engineering, and the challenges and limitations of AI, as well as future research directions. This analysis sheds light on the symbiotic relationship between human insight and artificial intelligence in software requirements engineering and navigates through the contributions, conclusions, and future directions suggested by the selected studies. Below, we highlight the importance of each aspect analyzed and provide a rationale as to why that aspect is essential to be analyzed in the context of this work:

1. **Effectiveness in Generating Requirements:** Evaluating ChatGPT's effectiveness in generating software requirements. We considered this characteristic to assess ChatGPT's ability to automate and optimize the elicitation and specification of requirements, which is traditionally an arduous process in terms of time and human effort.
2. **Role of Human Input:** Human input is fundamental to guiding and improving the quality of the requirements generated by ChatGPT. Human supervision and feedback are essential for the tools to produce requirements that are aligned with stakeholder needs.
3. **Exploring Prompt Engineering:** Input to language models is fundamental for adequate results. We understand that evaluating this aspect significantly influences the quality and relevance of ChatGPT responses in requirements engineering.
4. **AI Challenges and Limitations:** When considering generative AI for requirements specification, it is fundamental to recognize the challenges and limitations of tools such as ChatGPT. For example, issues such as bias, the possibility of generating hallucinations, and a lack of detail can affect the integrity of the generated requirements.
5. **Future Research Directions:** Any scientific work that proposes practical approaches should identify potential areas for future research and how studies can advance the use of ChatGPT in requirements engineering.
6. **Comparative Human Expertise:** Comparing human expertise is essential to understanding how ChatGPT aligns with or complements the knowledge and skills of human requirements engineers. This is particularly important because, although ChatGPT can automate specific processes, human expertise remains irreplaceable in understanding and analyzing the complex and contextual requirements of software projects.

Table 1. Comparative analysis of ChatGPT's impact on software requirements.

Study Reference (Author)	Publication Year	Effectiveness in Generating Requirements	Role of Human Input	Exploration of Prompt Engineering	AI Challenges and Limitations	Future Research Directions	Comparative Human Expertise
Liu et al. [2]	2022	✔				✔	
Abdelfattah et al. [14]	2023	✔				✔	
Arora et al. [15]	2023				✔	✔	
Belzner et al. [16]	2023				✔	✔	
Bencheik et al. [1]	2023	✔	✔			✔	
Cheng et al. [17]	2023			✔		✔	
El-Hajjami et al. [18]	2023	✔					
Kutzner et al. [19]	2023	✔					✔
Qian et al. [20]	2023		✔	✔		✔	
Ronanki et al. [3]	2023	✔				✔	✔
Rasheed et al. [21]	2023	✔		✔	✔		
Sridhara et al. [12]	2023	✔					
Subahi [22]	2023	✔					
Wang et al. [23]	2023	✔	✔		✔		
White et al. [6,7]	2023			✔		✔	
Zhang et al. [24]	2023	✔		✔		✔	

Table 1. *Cont.*

Study Reference (Author)	Publication Year	Effectiveness in Generating Requirements	Role of Human Input	Exploration of Prompt Engineering	AI Challenges and Limitations	Future Research Directions	Comparative Human Expertise
Fantechi et al. [25]	2024	✔				✔	✔
Luitel et al. [26]	2024	✔				✔	
Oswal et al. [27]	2024	✔		✔		✔	
Waseem et al. [28]	2024				✔	✔	✔
Yeow et al. [29]	2024	✔			✔	✔	

The highlighted issues are included because they provide a holistic view of using ChatGPT and other similar tools for requirements bidding, taking into account the benefits and potential challenges of this approach. Understanding these issues is critical to optimizing the use of ChatGPT to achieve requirements specifications.

5. Potential Benefits of Using ChatGPT

Based on the literature review, this section and the following one answer our Research Question: "How can ChatGPT be used to improve software requirements engineering processes, and what are the associated benefits, challenges, and ethical considerations?". From a requirements engineer's point of view, several potential benefits offered by ChatGPT have been identified. Table 2 lists these benefits and identifies each of the papers in the literature review where they are mentioned. In the following sections, we will discuss each of these in more detail.

Table 2. Potential benefits identified in the literature review.

Benefits	Paper
Improving Brainstorming and Idea Exploration	Bencheik et al. [1]
Continuous Learning	Abdelfattah et al. [14], Arora et al. [15], Belzner et al. [16], Rasheed et al. [21]
Minimize Human Error	Rasheed et al. [21]
Costs Savings	Rasheed et al. [21], Wang et al. [23]
Efficiency and Accuracy	Bencheik et al. [1], Waseem et al. [28]
Enhanced Productivity	Bencheik et al. [1], Liu et al. [2], Rasheed et al. [21], Yeow et al. [29]

5.1. Improving Brainstorming and Idea Exploration

Improving brainstorming is a significant benefit of incorporating ChatGPT into software requirements processes. ChatGPT helps generate a wide range of ideas and suggestions, stimulating creativity and exploring different possibilities. It offers different perspectives and draws from its extensive knowledge base, enriching the brainstorming process with new insights [1].

ChatGPT demonstrates proficiency in creative writing tasks by refining its output through various stages, such as brainstorming, creating stories or poems, and generating speeches. This means that ChatGPT is able to perfect brainstorming techniques to generate more creative and effective ideas for writing tasks [1].

ChatGPT inspires solutions and facilitates collaboration among team members with its ability to synthesize shared perceptions. Through iterative dialogue, ChatGPT allows ideas to be refined over time, resulting in broader requirements. In addition, ChatGPT provides real-time feedback on proposed ideas, helping to make informed decisions and ensure alignment with project goals.

5.2. Continuous Learning

Continuous learning is a valuable benefit of integrating ChatGPT into software requirements processes. ChatGPT continuously learns from interactions and feedback, adapting and improving to effectively capture evolving requirements. With access to vast sources of information, ChatGPT expands its knowledge base and provides more informed responses.

The use of real-time user feedback is a practice that contributes significantly to improving the quality of the software requirements refined by ChatGPT. Users can quickly correct errors or inaccuracies in the output generated by ChatGPT by immediately commenting on it. They can also provide additional details to further clarify the requirements [21]. Moreover, this real-time feedback can be used to enrich ChatGPT's training data. By exposing the model to a variety of interactions and corrections, the model can learn from this information and improve its ability to generate refined software requirements in the future [15].

In this iterative learning process, ChatGPT incorporates user feedback, which contributes to better documentation requirements by improving quality and accuracy. It develops an in-depth knowledge of specific domains and terminologies, leading to more appropriate responses to context-based information needs. These incremental learning practices positively impact the effectiveness of capturing and processing requirements requests, resulting in faster turnaround times. In addition, by staying current with emerging software industry trends and changing project needs, ChatGPT maintains its relevance as a helpful software requirements tool.

5.3. Minimize Human Error

The integration of ChatGPT into the software requirements engineering process significantly mitigates the risk of human error. ChatGPT's automation of documentation helps to reduce errors associated with manual transcription and interpretation. ChatGPT ensures a consistent approach to requirements documentation by conforming to established rules and standards, thereby minimizing variation due to human factors. The ChatGPT's ability to generate documentation with unambiguous and precise requirements avoids misunderstandings [7]. During the quality assurance phase, the model plays a critical role in identifying and resolving discrepancies and inconsistencies within the requirements, facilitating the correction of errors before they propagate through the development process.

Through cross-referencing and validation, ChatGPT ensures policy compliance and reduces the risk of errors. ChatGPT helps prevent errors from spreading throughout the requirements documentation by prompting for additional information or clarification when needed. Integrating ChatGPT results in more accurate and reliable documentation, easier software development, and fewer costly errors.

Thus, automating the process of creating software requirements can help reduce human error. In this way, ChatGPT's ability to detect inconsistencies in software requirements reduces the likelihood of human error in the requirements creation process [21].

5.4. Cost Savings

Gathering initial requirements for software projects involves stakeholder engagement, where ChatGPT interacts with stakeholders. Based on these interactions, it creates documentation of the software requirements. It's important to note that this is an example of how ChatGPT can help analyze and prioritize requirements. Processing large data volumes and identifying inconsistencies or gaps in the specification can be instrumental in ensuring quality assurance activities. Using ChatGPT for software requirements can save costs by streamlining the requirements gathering and analysis process, improving documentation quality, and increasing overall project management efficiency. These contributions result in cost savings in man-hours and reduced manual effort [21].

Also, ChatGPT helps generate code snippets from natural language descriptions in software engineering, increasing developer efficiency and allowing focus on higher-level design. Furthermore, it assists in debugging by detecting errors and providing suggestions for fixing them, speeding up the debugging process, and reducing some of the development time [30].

ChatGPT's software requirements generation process automation allows developers to focus on higher value tasks. At the same time, users only need to review the software requirements generated, which increases developer productivity. These factors help reduce development and labor costs [21,23].

5.5. Efficiency and Accuracy

One of the key benefits of integrating ChatGPT into software requirements processes is efficiency and accuracy. Stakeholder engagement, customer interviews, and feedback analysis are some use cases where ChatGPT allows faster and more comprehensive requirements gathering than traditional labor-intensive methods. Documentation helps with automation, leading to better accuracy and completeness while saving time and helping reduce human error. Another benefit of using ChatGPT is the ability to quickly analyze large amounts of data to identify patterns, inconsistencies, and gaps to ensure requirements are complete and meet stakeholder needs.

With its ability to identify inconsistencies and ambiguities in quality assurance, ChatGPT provides immediate suggestions for resolving them by ensuring that responses are concise, accurate, and relevant. Reducing the level of rework by preventing misunderstandings or omissions in software development projects contributes to improved efficiency and resource utilization throughout all phases [31].

The effectiveness and accuracy of ChatGPT throughout the software development process are very positive [28]. This is evidenced by the quality of the software requirements generated by ChatGPT, which are comparable to those generated by humans. However, it is important to emphasize the need for human expertise and feedback mechanisms to improve its performance [1].

5.6. Enhanced Productivity

The potential of ChatGPT to improve productivity in software development, especially in software requirements generation, has been widely recognized [1,21,29].

ChatGPT automates software engineering processes, including code development, testing, updating, and documentation, allowing human developers to focus on more creative problem-solving skills and innovation. This significantly increases the productivity of software development teams by allowing them to take on larger and more challenging projects. In addition, the ability of programmers to quickly turn their thoughts into code ensures that software is released faster and at a lower cost [2].

6. Limitations and Risks of Using ChatGPT

Despite its potential, the use of ChatGPT in software requirements engineering raises several concerns. Considering the literature reviewed, some limitations and risks are identified. Table 3 provides a list of these limitations and risks and identifies each of the papers in the literature review in which they were discussed. These concerns are presented in the following sections.

Table 3. Limitations and risks identified in the literature review.

Limitations and Risks	Paper
Addressing Bias in GenAI Systems	Bencheik et al. [1], El-Hajjami et al. [18], Luitel et al. [26], Ronanki et al. [3]
Information Hallucination	Belzner et al. [16], Qian et al. [20], Ronanki et al. [3]
Lack of Explainability and Transparency	Ronanki et al. [3]
Susceptibility to Attacks	Arora et al. [15]
Reasoning Errors	Arora et al. [15], Ronanki et al. [3]
Over-reliance	Arora et al. [15], Waseem et al. [28], Yeow et al. [29]
Transparency and Accountability	Belzner et al. [16]

6.1. Addressing Bias in GenAI Systems

GenAI systems are susceptible to incorporating biases present in their training data. This bias can lead to results that perpetuate discrimination or turn out to be misleading interpretations, which can lead to dissent and public disapproval at the political, social justice, and legal levels [30]. Manifestations of bias include the following:

- **Training data bias:** Language models learn from large datasets of human language. If these datasets contain biases related to race, gender, or socioeconomic status, the model may internalize these biases and reproduce them in its responses. A classic example might be that if there is a gender bias in the training data, the model is more likely to favor a particular gender in its responses.
- **User interaction bias:** Responses generated by chatbots are influenced by user input. The model can learn and perpetuate these biases if users consistently ask biased or prejudiced questions. Consequently, if users frequently ask discriminatory questions that target a particular group, the model may generate responses that reinforce these biases.
- **Algorithmic bias:** Bias can arise from the algorithms used to train and run language models and chatbots. For example, suppose a model is optimized to achieve accuracy or engagement as its performance metric. In this case, it may favor responses that meet that metric, even though those responses may also be biased.
- **Contextual bias:** Contextual bias is possible when chatbots generate responses based on contextual information provided by the user. The model could generate biased responses if factors such as language or location are biased. For example, if a user asks about a particular culture or religion, and the model has not been trained in that particular context, it could potentially provide biased responses due to limited knowledge.

Although ChatGPT provides good accuracy and efficiency in generating software requirements, there is still the possibility that the generated results contain biases [26]. Also, the use of specific data sets to train the model can lead to bias in the results [18]. To help mitigate these biases, increasing the model's training data is a viable solution. Furthermore, developing human supervision mechanisms to identify and correct incorrect requirements can also be an effective bias mitigation strategy [1,3,26].

It is also important to consider bias in GenAI systems so that the requirements and specifications generated by tools such as ChatGPT are based on an ethical and professional context.

6.2. Information Hallucination

ChatGPT can occasionally produce incorrect results, known as "hallucinations", which can lead to the generation of false-positive results [16]. Adopting human supervision mechanisms in the implementation of requirements generated by ChatGPT can help mitigate these hallucinations of the model and ensure a correct interpretation of the generated output [3].

When ChatGPT generates inaccurate and purely fictional information or hallucinations, these conditions occur when the model lacks sufficient knowledge or information and must guess and attempt to fill in the missing piece using training examples as reference points. Incorrect outputs can manifest as hallucinations, causing problems in applications where accuracy is critical [5].

Research is currently focused on understanding what makes hallucinations possible in language models. Recent studies have shown that this phenomenon may be multifaceted and related to factors such as model training techniques, data quality, and architectural design. Language models may also be biased to produce more "interesting" or well-written output, increasing the likelihood of hallucinations. Several methods have been proposed to address this issue, one of which is to use a large dataset that can help reduce the number of incorrect assumptions made by the model. This method exposes the model to different contexts and information during training, reducing the risk of hallucination.

Understanding the phenomenon of information hallucination through ChatGPT, where fictitious data can be generated, helps to establish protocols for fact-checking, which is essential for the reliability of requirements specifications and, by extension, the robustness of the resulting software.

The problem of the lack of explainability and transparency is raised in the literature, where its development is attributed to the lack of verifiability of GenAI systems, which will be discussed next.

6.3. Lack of Explainability and Transparency

The results of LLMs are more difficult to understand because they often require additional explanation, making it even more difficult to penetrate their inner meaning and trace their genesis and fidelity.

Explainability is one of the main limitations of using LLMs due to their immense number of parameters. For example, systems such as GPT-3 have 175 billion parameters, creating a highly complex network of interconnected nodes required for their operation. This complexity poses a significant problem for humans in understanding and interpreting the decision-making mechanisms implemented by the machine. In addition, training large language models requires collecting large amounts of data from multiple sources.

The patterns and correlations learned from the data lead to implicit biases and associations that may not be immediately obvious or interpretable. Accordingly, when a large language model makes a decision, it is difficult to isolate what caused that decision because it has many interrelated parts. This difficulty also explains why the results generated by the model cannot be articulated easily.

The main reason for the lack of explicability is that it may hinder the trust and full adoption of large language models (LLMs) in security applications. The inability to understand and decipher the decision-making mechanisms of LLMs may raise doubts about their trustworthiness, fairness, and potential biases. As a result, stakeholders such as policymakers, businesses, and users may require a longer period of time before they can rely on LLMs for essential contexts, prioritizing transparency and accountability [8].

ChatGPT works by using large neural networks that have been trained on large amounts of data and refined to perform specific natural language processing (NLP) tasks. Although the software requirements generated by ChatGPT can be similar in quality to those generated by humans, it can be difficult for users to understand the mechanisms that led ChatGPT to produce the result [3].

The need for explainability and transparency in LLM models such as ChatGPT is driving research to make GenAI more interpretable and reliable, which is essential for its adoption in critical software specification and other software engineering processes.

6.4. Susceptibility to Attacks

Requirements are fundamental for software engineering as they describe the functionalities, characteristics, and constraints of the system to be developed. They are essential to guiding the entire development process, from conception to delivery of the final product. Furthermore, requirements often contain sensitive information [15]. The disclosure of this sensitive information in software requirements can be explored with malicious intent through attacks on the system, raising security and privacy concerns. Therefore, it is critical to implement robust security measures to protect requirements and ensure the confidentiality and integrity of sensitive information [32].

Three categories of adversarial attacks could subject ChatGPT to misleading prompt injection, jailbreak attacks aimed at stealing sensitive information, and data poisoning methods, which would alter ChatGPT's output.

The main goal of such adversarial attacks is to disrupt or take control of the output of a large language model. Such attacks involve deliberately altering the input text at a small level so that LLM misinterprets it and returns inaccurate or harmful results. A typical example of an adversarial attack is a text injection attack. The perpetrator attacks by inserting well-crafted instructions into the input, which the LLM interprets as commands. For example, such injected text could be "delete all files" fed into an LLM controlling a computer system. As a result, the LLM could treat this input as a command and delete everything from the system according to the injected text [32].

The detection and mitigation of adversarial attacks on ChatGPT highlight the need to strengthen the security of GenAI systems, which is fundamental to ensuring the integrity, confidentiality, and cohesion of the requirements generated.

6.5. Reasoning Errors

According to [33], even large language models (LLMs) such as ChatGPT are sometimes misled by ambiguities in the question or poor understanding of complex logical operations. They need to acquire planning and reasoning skills, and also have limited awareness and common sense about the world.

When the generation of ambiguous requirements occurs, human intervention is necessary to avoid misinterpretations of these requirements. Ambiguous requirements can lead to a scenario of uncertainty in the decision-making process, which is undesirable for developers [3]. Human intervention can help clarify ambiguities, identify gaps, and ensure that requirements are clearly defined and understood by all parties involved in the software development process [15].

The detection of reasoning errors by LLMs is driving efforts to improve GenAI's reasoning and logic capabilities, which are essential for the accurate generation of software requirements.

6.6. Over-Reliance

Replacing human expertise with GenAI tools can overlook critical aspects of requirements gathering, such as understanding contextual nuances and implicit needs [15]. Despite all the different challenges that GenAI faces in its integration into software engineering requirements, the challenge of trust remains prominent, as highlighted in the research [29].

One of the main pitfalls of using ChatGPT for software requirements is the temptation to rely too much on the generated text without sufficient verification. This over-reliance on AI can lead to a lack of trust among stakeholders [15]. While ChatGPT can produce coherent and seemingly relevant responses, it requires more contextual understanding and may need to provide more accurate or complete information. To mitigate this lack of contextual knowledge, users should provide clear and detailed input prompts, offer contextual information where and when needed, and refine requirements based on ongoing dialogue and clarification. Therefore, developers must exercise caution and critically evaluate the suitability and accuracy of the generated content in the context of specific software requirements [31].

Identifying the risk of over-reliance on ChatGPT highlights the need to combine artificial intelligence with human expertise, promoting a balance that can lead to more sophisticated software development practices that are responsive to human needs.

6.7. Transparency and Accountability

Establishing transparency in the use of GenAI technologies, including accountability for the results produced and the steps taken to avoid bias, is essential. The ChatGPT algorithm operates as a black box model, providing no visibility or explanation to end users.

The lack of transparency can become an obstacle to responsible oversight, making it difficult to detect and correct problems such as bias, errors, or adverse outcomes in output requirements. Mechanisms such as transparency and accountability are essential for stakeholders to trust and understand, rather than doubt or resist, the decisions made by ChatGPT [30].

It is important to maintain human responsibility for the results generated. When the potential consequences of errors resulting from incorrect use of LLMs are significant, human oversight is critical. While the results generated by automation are not completely reliable, human analysis and review of these results not only ensures their quality, but also incorporates human responsibility, adding a crucial layer of accountability and security [16].

Establishing transparency and accountability in the use of GenAI technologies such as ChatGPT reinforces the importance of ethics in software engineering. It also promotes technological development that respects human rights and values and considers social and ethical aspects for each application context.

6.8. Summary of the Limitations and Risks of Using ChatGPT

By addressing these challenges and risks, the scientific community and professionals in the field can develop more robust and ethically responsible practices for integrating ChatGPT and other GenAI tools into software requirements engineering. This will improve the development process and contribute to the evolution of knowledge and practices in software engineering [30].

7. Future Research Directions

Given the transformative potential and limitations identified in the use of ChatGPT in software requirements engineering, we propose future research focused on the following critical areas:

- Prompt generation: New prompt construction techniques suitable for each stage of the software development lifecycle need to be explored. This will result in clearer and more comprehensive requirements. Experimentation with different prompt construction strategies can improve the quality of the generated requirements.
- Mitigate bias and increase equity: Investigate methods for identifying, measuring, and mitigating bias in ChatGPT-generated software requirements engineering. This includes developing techniques for training on more balanced datasets and exploring the implementation of fairness-aware algorithms for software requirements engineering.
- Suggest ways of clarifying the decision-making processes to improve clarity and transparency: Possible ways include more interpretable models or the development of tools to provide insight into model reasoning.
- Reliable data handling and processing protocols that address data privacy and security concerns: Research could explore encryption methods, differential privacy techniques, and secure mechanisms to ensure the secure storage of sensitive information.
- Improve ChatGPT's error detection and correction capabilities, focusing more on logical and factual inaccuracies: This could involve using external knowledge bases, improved model training techniques, or a human-in-the-loop verification system.
- Strategies to minimize overreliance and maintain a well-orchestrated interaction between GenAI and human intellect in problem-solving: We suggest defining and setting standards for the use of ChatGPT in software requirements engineering, how GenAI can assist in performing specific tasks, and how human oversight can be emphasized.
- Resistance to adversarial attacks: It is necessary to study ways to improve ChatGPT's ability to resist counterattacks and malicious manipulation in order to promote reliability and consistency in terms of system requirements.
- Ethical and social issues: The use of ChatGPT may raise ethical issues regarding privacy breaches and redundancy. We recommend in-depth research into the two main issues causing unemployment and increasing economic inequality, and promoting equal access to GenAI tools.

8. Conclusions

This paper has evaluated the incorporation of ChatGPT into software requirements engineering, identifying promising benefits and significant challenges. We also explored the usefulness of ChatGPT as a facilitator of iterative development and its use in capturing changing project requirements.

ChatGPT's ability to enhance brainstorming, reduce human error, save costs, and improve efficiency and accuracy highlights its potential to transform software requirements engineering practices. However, critical issues such as bias, information hallucination, lack of explainability, vulnerability to attacks, reasoning errors, overreliance, and ethical considerations underscore the need for cautious and informed implementation.

We propose future research to address these challenges in order to responsibly realize ChatGPT's full potential. By focusing on mitigating bias, improving explainability, ensuring data privacy, correcting errors, and understanding ethical implications, we can move toward a more effective and ethical integration of ChatGPTin software development.

In summary, as we move forward, it is critical to adopt a collaborative approach that leverages ChatGPT capabilities and human expertise to improve the quality and reliability of software requirements.

Author Contributions: Conceptualization, R.R.S. and J.B.; methodology, J.B. and R.R.S.; software, N.M.; validation, J.B., R.R.S. and N.M.; formal analysis, J.B. and R.R.S.; investigation, N.M.; resources, R.R.S. and J.B.; data curation, N.M.; writing—original draft preparation, N.M.; writing—review and editing, J.B. and R.R.S.; visualization, J.B., R.R.S. and N.M.; supervision, J.B. and R.R.S.; project administration, J.B. and R.R.S.; funding acquisition, J.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bencheikh, L.; Höglund, N. Exploring the Efficacy of Chatgpt in Generating Requirements: An Experimental Study. Bachelor's Thesis, Chalmers University of Technology, Göteborg, Sweden, 2023.
2. Liu, K.; Reddivari, K. Artificial Intelligence in Software Requirements Engineering: State-of-the-Art. In Proceedings of the IEEE 23rd International Conference on Information Reuse and Integration for Data Science (IRI), San Diego, CA, USA, 9–11 August 2022.
3. Ronanki, K.; Berger, C.; Horkoff, J. Investigating ChatGPT's Potential to Assist in Requirements Elicitation Processes. In Proceedings of the 49th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Durres, Albania, 6–8 September 2023.
4. Chang, Y.; Wang, X.; Wang, J.; Wu, Y.; Yang, L.; Zhu, K.; Chen, H.; Yi, X.; Wang, C.; Wang, Y.; et al. A Survey on Evaluation of Large Language Models. *ACM Trans. Intell. Syst. Technol.* **2023**, *15*, 1–45. [CrossRef]
5. Fan, A.; Gokkaya, B.; Harman, M.; Lyubarskiy, M.; Sengupta, S.; Yoo, S.; Zhang, J.M. Large language models for software engineering: Survey and open problems. *arXiv* **2023**, arXiv:2310.03533.
6. White, J.; Hays, S.; Fu, Q.; Spencer-Smith, J.; Schmidt, D.C. ChatGPT Prompt Patterns for Improving Code Quality, Refactoring, Requirements Elicitation, and Software Design. *arXiv* **2023**, arXiv:2303.07839.
7. White, J.; Fu, Q.; Hays, S.; Sandborn, M.; Olea, C.; Gilbert, H.; Elnashar, A.; Spencer-Smith, J.; Schmidt, D.C. A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv* **2023**, arXiv:2302.11382.
8. Nguyen-Duc, A.; Cabrero-Daniel, B.; Przybylek, A.; Arora, C.; Khanna, D.; Herda, T.; Rafiq, U.; Melegati, J.; Guerra, E.; Kemell, K.K.; et al. Generative Artificial Intelligence for Software Engineering—A Research Agenda. *arXiv* **2023**, arXiv:2310.18648.
9. Westfall, L. Software Requirements Engineering: What, Why, Who, When, and How. *Softw. Qual. Prof.* **2005**, *7*, 17.
10. Marr, B. A Short History of ChatGPT: How We Got to Where We Are Today. Available online: https://www.forbes.com/sites/bernardmarr/2023/05/19/a-short-history-of-chatgpt-how-we-got-to-where-we-are-today/?sh=454c1d75674f (accessed on 1 March 2024).
11. Hörnemalm, A. ChatGPT as a Software Development Tool: The Future of Development. Master's Thesis, Umeå University, Faculty of Science and Technology, Umeå, Sweden, 2023. Available online: https://urn.kb.se/resolve?urn=urn:nbn:se:umu:diva-209909 (accessed on 1 March 2024).
12. Sridhara, G.; Mazumdar, S. ChatGPT: A Study on its Utility for Ubiquitous Software Engineering Tasks. *arXiv* **2023**, arXiv:2305.16837.
13. Halevi, G.; Moed, H.; Bar-Ilan, J. Suitability of Google Scholar as a source of scientific information and as a source of data for scientific evaluation—Review of the Literature. *J. Informetr.* **2017**, *11*, 823–834. [CrossRef]
14. Abdelfattah, A.M.; Ali, N.A.; Elaziz, M.A. Roadmap for Software Engineering Education using ChatGPT. In Proceedings of the 2023 International Conference on Artificial Intelligence Science and Applications in Industry and Society (CAISAIS), IEEE, Galala, Egypt, 3–5 September 2023.
15. Arora, C.; Grundy, J.; Abdelrazek, M. Advancing Requirements Engineering through Generative AI: Assessing the Role of LLMs. *arXiv* **2023**, arXiv:2310.13976.

16. Belzner, L.; Gabor, T.; Wirsing, M. Large Language Model Assisted Software Engineering: Prospects, Challenges, and a Case Study. In Proceedings of the International Conference on Bridging the Gap between AI and Reality, Crete, Greece, 23–28 October 2023.
17. Cheng, Y.; Chen, J.; Huang, Q.; Xing, Z.; Xu, X.; Lu, Q. Prompt sapper: A LLM-empowered production tool for building AI chains. *ACM Trans. Softw. Eng. Methodol.* **2023**. [CrossRef]
18. El-Hajjami, A.; Fafin, N.; Salinesi, C. Which AI Technique Is Better to Classify Requirements? An Experiment with SVM, LSTM, and ChatGPT. *arXiv* **2023**, arXiv:2311.11547.
19. Kutzner, T.; Gröpler, J. Supporting Students in the Creation of Requirements and Functional Specifications in Interdisciplinary Software Development Projects with the Help of AI-Based Text Generation Tools. X Jornadas Iberoamericanas de Innovación Educativa en el Ámbito de las TIC y las TAC. 2023. Available online: https://accedacris.ulpgc.es/bitstream/10553/128281/1/Supporting_students_creation.pdf (accessed on 19 April 2024).
20. Qian, C.; Cong, X.; Liu, W.; Yang, C.; Chen, W.; Su, Y.; Dang, Y.; Li, J.; Xu, J.; Li, D.; et al. Communicative agents for software development. *arXiv* **2023**, arXiv:2307.07924.
21. 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* **2023**, arXiv:2311.18440.
22. Subahi, A.F. BERT-Based Approach for Greening Software Requirements Engineering Through Non-Functional Requirements. *IEEE Access* **2023**, *11*, 103001–103013. [CrossRef]
23. Wang, Z.; Li, J.; Li, G.; Jin, Z. ChatCoder: Chat-based Refine Requirement Improves LLMs' Code Generation. *arXiv* **2023**, arXiv:2311.00272.
24. Zhang, J.; Chen, Y.; Niu, N.; Liu, C. A preliminary evaluation of chatgpt in requirements information retrieval. *arXiv* **2023**, arXiv:2304.12562.
25. Fantechi, A.; Gnesi, S.; Semini, L. Exploring LLMs' Ability to Detect Variability in Requirements. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*; Springer Nature: Cham, Switzerland, 2024.
26. Luitel, D.; Hassani, S.; Sabetzadeh, M. Improving requirements completeness: Automated assistance through large language models. *Requir. Eng.* **2024**, *29*, 73–95. [CrossRef]
27. Oswal, J.U.; Kanakia, H.T.; Suktel, D. Transforming Software Requirements into User Stories with GPT-3.5-: An AI-Powered Approach. In Proceedings of the 2024 2nd International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT), IEEE, Bengaluru, India, 4–6 January 2024.
28. Waseem, M.; Das, T.; Ahmad, A.; Liang, P.; Fahmideh, M.; Mikkonen, T. ChatGPT as a Software Development Bot: A Project-based Study. *arXiv* **2024**, arXiv:2310.13648.
29. Yeow, J.S.; Rana, M.E.; Majid, N.A.A. An Automated Model of Software Requirement Engineering Using GPT-3.5. In Proceedings of the 2024 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems (ICETSIS), Al Ekir, Kingdom of Bahrain, 28–29 January 2024.
30. Fraiwan, M.; Khasawneh, N. A Review of ChatGPT Applications in Education, Marketing, Software Engineering, and Healthcare: Benefits, Drawbacks, and Research Directions. *arXiv* **2023**, arXiv:2305.00237.
31. Hariri, W. Unlocking the Potential of ChatGPT: A Comprehensive Exploration of its Applications, Advantages, Limitations, and Future Directions in Natural Language Processing. *arXiv* **2023**, arXiv:2304.02017.
32. Kalla, D.; Kuraku, S. Advantages, Disadvantages and Risks associated with ChatGPT and AI on Cybersecurity. *J. Emerg. Technol. Innov. Res.* **2023**, *10*, h84–h94.
33. Zhang, Q.; Zhang, T.; Zhai, J.; Fang, C.; Yu, B.; Sun, W.; Chen, Z. A Critical Review of Large Language Model on Software Engineering: An Example from ChatGPT and Automated Program Repair. *arXiv* **2023**, arXiv:2310.08879.