Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X Volume 4, Issue 4, 2026

A Comparative Analysis of Human vs LLM-Generated Software Requirements: Evaluating Quality, Ambiguity, and Completeness in AI-Assisted Requirement Engineering

Sadia Shoukat
MS in Software Engineering, Iqra University Islamabad Campus
sadiashoukat064@gmail.com

DOI: https://doi.org/

Keywords
Requirement Engineering, Large Language Models (LLMs), Artificial Intelligence in Software Engineering, Software Requirement Quality, Human-Computer Interaction, Ambiguity Detection, Software Completeness, Empirical Software Engineering, AI-Assisted Requirement Engineering, Comparative Analysis

Article History
Received on 25 Mar 2026
Accepted on 12 April 2026
Published on 30 April 2026

Corresponding Author: *

Sadia Shoukat
Email:sadiashoukat064@gmail.com

The adoption of Large Language Models (LLMs) for software development has profoundly impacted software requirement engineering. While the use of LLMs in generating software requirements is increasing, there is a need for more research on the effectiveness of LLM-generated software requirements. This research compares requirements generated by humans and LLM to assess their quality in terms of understanding, ambiguity and completeness. The research uses a controlled experimental approach to generate requirements for several software use cases by humans and an LLM. These requirements are then measured against established criteria and analyzed by experts and statistical methods. The results show that although the LLM-generated requirements are efficient and consistent, they are also more ambiguous and less complete than human-generated requirements. In contrast, human-generated requirements are found to be more situational and accurate, but less consistent. This research offers an evaluation framework for AI-assisted requirement engineering, and suggests the benefits of a hybrid human-AI approach. Our findings offer insights into enhancing the effectiveness of AI-assisted software development.

INTRODUCTIO

Background of Requirement Engineering

It is well acknowledged that Requirement Engineering (RE) is an important stage in the software development process as it lays the foundation for system design, development and evaluation. Numerous studies have reported that the root cause of software project failure is related to inadequate requirements (Sommerville, 2016). Software requirements are expected to be clear, complete, consistent, and unambiguous, in order to ensure effective communication between stakeholders, and avoid confusion (Wiegers & Beatty, 2013).

However, the RE process is complex and prone to error. Software requirements are often described using natural language, which can be vague and ambiguous. Moreover, factors such as diverse stakeholder perspectives, the dynamic nature of user requirements, and pressure to complete requirements within a time-frame can further complicate the elicitation and specification process. These factors are known to be major sources of defects, rework and cost in software development (Boehm & Basili, 2001). As a result, improving the quality and dependability of software requirements remains a critical area of research and practice.

Rise of Large Language Models in Software Engineering

The rapid progress in Artificial Intelligence (AI) has resulted in the emergence of Large Language Models (LLMs) with impressive text generation capabilities and support for a range of knowledge-based tasks. LLMs developed by OpenAI and other companies have been effectively used in areas like code generation, automated documentation, and software testing (Brown et al., 2020; Nijkamp et al., 2023).

LLMs' incorporation in software engineering has brought new possibilities for automating and supporting requirement engineering tasks. LLMs have been proposed to be used to help draft initial requirement documents, to collaboratively refine user stories, and to assist in documentation in Agile and DevOps approaches (Ahmad et al., 2022). These features can save time, labor, and enhance productivity.

But using LLMs for requirement engineering is not without risks. As a result of their probabilistic nature of language modeling, LLMs may produce plausible but inaccurate or incomplete results. It has been reported that they may introduce ambiguity, neglect crucial requirements or not adequately capture domain-specific details (Bommasani et al., 2021). As such, although LLMs have several advantages, their potential in generating quality software requirements is still unclear.

1.3 Research Gap

While the use of AI in software engineering has been widely explored, research on the application of LLMs in software requirement engineering is still in its infancy. Most research has centered on code generation, code testing and code documentation, with less attention paid to the assessment of requirement quality (Pearce et al., 2025). There is a shortage of unified evaluation methods for evaluating the quality of LLM-generated requirements. Specifically, there is a lack of empirical research that evaluates the difference between human-generated and LLM-generated requirements based on established quality criteria, such as correctness, ambiguity, and completeness (Cheng et al., 2024). This prevents a full picture of the potential and limits of LLMs in requirement engineering.

Thus, there is a research gap in the need for a systematic and empirical assessment of the quality of software requirements generated by LLMs compared to human-generated ones.

Problem Statement

While LLMs are increasingly being used in software development, the effectiveness of LLM-generated software requirements has yet to be rigorously evaluated. It is unclear if the requirements are of sufficient quality for system development. Inconsistency, incompleteness and contextual errors continue to be a major problem, which hampers the use of LLMs in requirement engineering.

Research Objectives

This research aims to:

To compare the quality, ambiguity and completeness of human and LLM-generated software requirements.

To provide empirical evidence on the use, effectiveness and limitations of LLMs in the context of requirement engineering.

Research Contributions

This study makes the following contributions:

A framework for evaluating the quality of software requirements in the context of AI assistance is developed.

A quantitative comparison of human and LLM-generated requirements using established metrics is performed.
Critical aspects of LLM-generated requirements, such as ambiguity and completeness, are highlighted.
Guidelines for enhancing AI-assisted requirement engineering are offered.

Literature review

Role of Requirement Engineering in Software

Requirement Engineering (RE) has been well recognized as a key activity in software engineering that deals with the process of identifying, analyzing, documenting and managing software requirements. There have been many reports that mistakes made in the requirement phase flow through the software development process and impact the quality, cost and maintainability of the system (Sommerville, 2016). Consequently, requirements accuracy and correctness are regarded as a critical factor for project success.
Software requirements are required to meet various quality characteristics such as clear, consistent, complete and unambiguous. They are deemed critical for minimizing potential stakeholder misunderstandings and to assure that the software system developed meets user expectations (Wiegers & Beatty, 2013). But the requirement elicitation process is not straightforward because of the presence of a diverse range of stakeholders with different viewpoints.
The use of natural language for requirement documentation adds to these challenges as it is inherently ambiguous and variable. It has been noted that small inconsistencies in requirement statements can cause major problems in the development phase (Boehm & Basili, 2001). These issues are further exacerbated in large and distributed software development due to communication gaps and geographical distances.

Issues in Traditional Requirement Engineering

Human-centered processes are at the heart of traditional Requirement Engineering processes, which leads to variability in requiring. There are reports that stakeholders often face difficulties in specifying requirements in complete and clear forms, and these require further clarification and refinement in development iterations. This slows down the development process and drives up the cost.
In Agile development, it is accepted that requirements will evolve; however, changes in requirements may result in inconsistencies in documentation and software design. Inconsistency in Agile requirement documentation has been pointed out as one of the reasons for misunderstanding and confusion (Boehm & Basili, 2001). The nature of distributed software development brings in communication issues that impact requirement quality. Geographical distance between teams can lead to unclear and inconsistent requirements. These challenges reveal the shortcomings of existing human-centered approaches in requirement engineering and suggest the need for more formal and automated approaches.

Artificial Intelligence in Software Engineering

Artificial Intelligence (AI) has been used in software engineering to boost productivity, automate processes, and facilitate decision making. Machine learning has been used for defect prediction, testing and software maintenance. In recent years, developments in deep learning and natural language processing have led to the ability of AI to understand and generate natural language text.
In particular, Large Language Models (LLMs) have shown impressive language processing abilities. Models trained by OpenAI and others on massive amounts of data can complete text, summaries content, generate code and draft requirements (Brown et al., 2020). AI-assisted tools have been found to enhance productivity for documentation and coding. AI-based tools are also being used in Agile and DevOps practices to aid in fast prototyping and sprint-based development (Ahmad et al., 2022). But recent literature has also debated issues of transparency and trust with AI.

Large Language Models in Requirement Engineering

Using Large Language Models in Requirement Engineering is an emerging field. LLMs have been used to create user stories, requirements, and specifications in response to natural language inputs. It has been found that these models can often generate outputs that are grammatically correct and contextually appropriate (Nijkamp et al., 2023).
However, there have been reported shortcomings in LLMs' semantic comprehension. These models rely on statistical learning rather than understanding to generate their responses, which may not always be accurate in the domain. It has been observed that

LLMs may hallucinate information, where the generated requirements may seem plausible but do not reflect the true system requirements (Bommasani et al., 2021). Research has also shown that LLM-generated requirements may not be comprehensive, especially in large systems that require domain-specific expertise. LLMs' lack of explicit reasoning leads to potential missing functional or non-functional requirements. Additionally, there is a risk of ambiguity in generated requirements, particularly in critical safety systems.

Requirements Generated by Humans

The need for humans has been a fundamental part of Requirement Engineering because of their ability to provide domain knowledge, understand context and negotiate with stakeholders. Humans can interpret implicit requirements and negotiate with stakeholders to reach agreement and compromise.

But human-produced requirements have their drawbacks. There is evidence that human-generated requirements are not consistent in structure and detail. Differences in writing styles, interpretations, and domain knowledge can result in unclear requirements. Further, biases in perception and communication barriers between stakeholders may also impact requirement quality. These constraints are exacerbated in a distributed development context with a lack of direct interaction. Some reports indicate that distributed development teams face challenges in establishing a shared understanding of requirements due to variations in interpretation and requirement documentation. These issues raise the need for tools to help human analysts write more standardized requirements.

Comparative Studies in Requirement Engineering

There have been many comparative studies between automated and human processes in software engineering. Historically, the focus has been on trade-offs between accuracy and ease-of-use between formal and informal specification methods. In recent years, there has been a focus on comparing AI-driven tools against conventional human practices.

Pearce et al. (2025) studied the application of AI tools in software engineering and found that although productivity gains were made, correctness and security problems were also detected. Likewise, other research has shown that while AI-generated code is fluent, it can lack contextual knowledge. Recent systematic reviews of generative AI in software engineering have indicated that current research has primarily been exploratory, with little empirical evidence on the effectiveness of AI-generated artifacts (Cheng et al., 2024). Furthermore, it has been observed that many studies are on code generation rather than requirement engineering, with a lack of comparative assessment at the requirement level.

Assessing Requirement Quality

The quality of requirements is usually measured along several dimensions, such as clarity, completeness, consistency and ambiguity. In particular, the requirement quality attributes of ambiguity and completeness are considered as the most important ones, since they affect system correctness and system usability (Wiegers & Beatty, 2013).

Ambiguity is the occurrence of multiple interpretations of a requirement statement. Ambiguity has been known to result in incorrect interpretation of requirements during design and implementation. Completeness is related to the inclusion of all functional and non-functional requirements in a specification. Different methods have been suggested to assess the quality of requirements, such as expert reviews, automated natural language processing, and machine learning for classifying textual requirements. But there is a lack of systematic approaches to evaluate and compare requirements written by humans and AI, especially in empirical studies.

Current Research in AI-Assisted Requirement Engineering

In recent times, AI-assisted Requirement Engineering has explored using machine learning models to assist in requirement elicitation and analysis. These tools seek to automate the requirement generation process by generating preliminary requirement drafts from stakeholder or system descriptions. Research has shown that AI-assisted systems can enhance the efficiency of requirement documentation by offering templates and requirement suggestions. But it has also been found that they may not be flexible enough for certain complex or domain-specific scenarios.

Studies have also indicated that human-AI collaboration may lead to better results than either fully automated or fully manual documentation approaches. These hybrid systems generate drafts

from AI systems, which are then refined and verified by humans.

Identified Research Gaps

A review of the literature reveals a number of research gaps in the area of AI-supported Requirement Engineering. While substantial progress has been achieved in the use of LLMs in software development, there is a lack of research on using them for the generation of requirements. The exploration of LLMs in requirement engineering has largely been limited to general assessments rather than specific evaluation in requirement engineering. Moreover, there is a lack of empirical evidence on the effectiveness of human-generated and LLM-generated requirements. The current body of research does not use well-established metrics and experiment designs to assess the quality of requirements. Moreover, few studies have provided a holistic evaluation of different quality aspects such as ambiguity, completeness and the overall quality of requirements.

These observations suggest that there is a need for empirical studies that rigorously investigate the use of LLMs for requirement engineering tasks and their effectiveness compared to human-produced requirements.

Methodology

Research Design

The current study is designed as a quantitative empirical comparative study to assess the impact of difference between human and Large Language Model (LLM) generated software requirements. This research follows the empirical software engineering approach, in which data is observed and collected to draw conclusions about software artifacts. An empirical experiment has been set up where two groups of requirement sources are investigated under the same controlled conditions. The sources in the first group are requirements generated by human participants familiar with software engineering terminology, whereas the second group are requirements generated by Large Language Model-based AI system. This setting allows us to compare human learning and reasoning with AI-based generative processes in requirement engineering.

The experiment is designed in such a manner that both groups are given the same system descriptions to ensure that the variations in the results are due to the generation source rather than the input. This ensures fairness of the experiment and increases the internal validity of the study.

Experimental Setting

The study has been carried out with standard scenarios of software systems that are representative of typical software systems encountered in software engineering. These range from a Library Management System, an Online Shopping System, a Student Information System to a Hospital Management System. These have been chosen because they are commonly used in requirement engineering research, and have a wide range of functional and non-functional requirements.

In the automated requirement generation process using AI, we have used a Large Language Model (LLM) created by OpenAI. We have used natural language instructions to guide the model for consistent generation. Human requirements have been collected from individuals with some exposure to software engineering principles, in order to provide a common ground of domain knowledge. All sets of requirements have been captured in a consistent format for evaluation. We have ensured that data presentation does not affect evaluation results.

Data Generation Process

The generation of data has been conducted in a structured way to ensure it is repeatable. First, the description of each software system scenario has been provided in a prompt format. These prompts have been given to human participants and the LLM on the same grounds. The human participants have been asked to develop a comprehensive set of software requirements, including functional and non-functional requirements, based on the system descriptions. They have been asked to formulate requirements in natural language form as is common in requirement engineering documents.

At the same time, the system descriptions have been fed to the LLM, with instructions to produce the same requirement sets. The prompts have been designed to be clear and to prompt the LLM to generate requirement statements similar to those of humans.

All sets of requirements have been gathered and grouped into datasets for each set of sources. These have been made ready for evaluation.

Requirement Evaluation Framework

A multi-dimensional evaluation framework has been adopted for the evaluation of software requirements. The framework is built on the foundations of the literature in requirement engineering and software quality evaluation.

The quality of requirements has been assessed using three key dimensions: quality, ambiguity and completeness. These have been chosen because they are of significant importance in establishing the success of software requirements in real-world development.

Requirement quality has been defined as the degree to which a requirement statement is correct, clear and relevant. The correctness of a requirement has been defined in terms of the accuracy of its representation of system functionality and clarity of its articulation.

Requirement ambiguity has been assessed based on whether the requirement statement has more than one possible interpretation. Requirements that use unclear terms, have unclear relationships or have multiple meanings have been deemed extremely ambiguous. The completeness of requirements has been evaluated based on how well the system's functional and non-functional requirements have been addressed. If features are missing or specifications are incomplete, they have been considered to have low completeness. The quality of each requirement has been assessed on a scale of 1-5, with 1 being very poor and 5 being excellent quality. This scale has been used for all aspects of the evaluation.

Evaluation Procedure

The evaluation has been conducted by domain-experienced evaluators knowledgeable in software engineering. All sets of requirements have been evaluated individually to avoid bias. A rubric has been designed to guide the evaluators in scoring the samples.

Evaluation has focused on individual requirement statements, rather than on the whole set. This approach has been used to enable a fine-grained comparison of human vs. LLM generated requirements.

To mitigate evaluator bias, the sources of requirements have been anonymized. The evaluators have not been provided with information about human versus LLM origin.

Data Analysis Techniques

We have used descriptive and inferential statistical methods to analyses the data collected from the evaluations. Central tendencies and measures of dispersion have been calculated for the two groups using descriptive statistics. The average and variance of the scores have been computed for each of the evaluation criteria, providing a first-level comparison of trends.

An independent sample t-test has been used to assess the statistical significance of differences between human-generated and LLM-generated requirements. This test has been chosen for its appropriateness in comparing the means of two unpaired sets.

The hypothesis test has been formulated as follows:

Null hypothesis (H_0): The quality, ambiguity, and completeness of software requirements generated by humans and LLMs do not differ significantly.

Alternative hypothesis (H_1): There is a significant difference between the two groups in terms of the evaluation dimensions.

A p-value of $p < 0.05$ has been used for statistical significance.

Results and Data Analysis

Overview of Collected Data

A total of four software system scenarios were used to generate requirement sets from both human participants and a Large Language Model (LLM). Each requirement set was evaluated using three predefined metrics: requirement quality, ambiguity, and completeness.

The evaluation scores were recorded on a five-point Likert scale, where higher values represent better performance. The collected data were organized and averaged to allow comparison between the two groups.

Descriptive Analysis

To simplify the analysis, mean (average) **scores** were calculated for each evaluation metric. The mean was computed using the standard formula:

$$\text{Mean} = \frac{\text{Sum of all scores}}{\text{Number of observations}}$$

Requirement Quality

The scores obtained for requirement quality were averaged across all scenarios.

Human-generated:

Mean = (4.5 + 4.2 + 4.3 + 4.3) / 4 = **4.32**

LLM-generated:

Mean = (3.8 + 3.6 + 3.9 + 3.8) / 4 = **3.78**

It was observed that human-generated requirements achieved a higher average score compared to LLM-generated requirements.

Requirement Ambiguity

Ambiguity was evaluated in terms of clarity (higher score = less ambiguity).

Human-generated:

Mean = (4.2 + 4.0 + 4.1 + 4.1) / 4 = **4.10**

LLM-generated:

Mean = (3.4 + 3.3 + 3.5 + 3.2) / 4 = **3.35**

The results indicate that LLM-generated requirements exhibited comparatively higher ambiguity.

Requirement Completeness

Completeness scores were calculated as follows:

Human-generated:

Mean = (4.3 + 4.1 + 4.4 + 4.2) / 4 = **4.25**

LLM-generated:

Mean = (3.6 + 3.5 + 3.7 + 3.6) / 4 = **3.60**

Human-generated requirements demonstrated more comprehensive coverage of system features.

Comparative Analysis

A direct comparison of average scores across all metrics shows consistent differences between the two groups:

Human-generated requirements scored higher in quality, clarity, and completeness

LLM-generated requirements showed **lower scores**, particularly in ambiguity and completeness

The difference between the groups can be expressed as:

Quality difference = 4.32 − 3.78 = **0.54**

Ambiguity difference = 4.10 − 3.35 = **0.75**

Completeness difference = 4.25 − 3.60 = **0.65**

These differences indicate that human-generated requirements outperform LLM-generated requirements across all evaluation dimensions.

Observed Patterns

During the evaluation process, certain patterns were identified in both types of requirement generation.

Human-generated requirements were observed to include more detailed and context-specific descriptions. These requirements demonstrated better alignment with system objectives and included explicit functional and non-functional aspects.

In contrast, LLM-generated requirements were generally well-structured and grammatically consistent. However, instances of generalized statements and lack of specificity were observed. Some requirement sets also showed minor omissions of critical system features.

Interpretation of Results

The calculated averages indicate that human-generated requirements consistently achieve higher scores across all evaluation metrics. The differences in scores suggest that while LLMs are capable of generating coherent requirement statements, limitations remain in terms of clarity and completeness.

The results also suggest that LLM-generated requirements may benefit from human refinement to improve accuracy and contextual relevance. The performance gap observed in ambiguity scores indicates that LLM outputs may require additional validation before being used in real-world software development.

Discussions

Understanding the Key Observations

The findings from the comparative study suggest that there are significant differences between human-generated and LLM-generated software requirements in terms of the three factors considered, that is, quality, ambiguity and completeness. It has been noticed that the mean scores for human-generated requirements were higher, implying that these requirements were better at capturing system requirements in a clear and contextual manner.

This is likely due to the capacity of human analysts to draw on contextual cues, domain expertise and iterative refinement. These results are consistent with the analysis of Sommerville (2016) where requirement engineering is considered a knowledge-based activity that involves human decision-making and stakeholder engagement.

Conversely, requirement statements generated by LLMs, while syntactically coherent and grammatically accurate, scored less highly. This implies while LLMs can generate syntactically correct requirements, there are still challenges in their capacity to provide semantically accurate and relevant requirements. These findings align with those of Bommasani et al. (2021), who report that language models often function through the use of probabilistic patterns.

Ambiguity of LLM-Generated Requirements

This study found that LLM-generated requirements were more ambiguous. This suggests that the requirements were more likely to contain ambiguous and vague statements. This is likely due to the nature of language model responses which are more generalized and prefer fluency.

It has been documented in previous research that ambiguous requirements can be misinterpreted at the time of system design and development (Wiegers & Beatty, 2013). The use of ambiguous words in LLM-generated responses indicates that LLMs may not always uphold the requirement for clarity in specification. Also, the lack of a clarification process in LLMs is another factor. Whereas human analysts can ask stakeholders for clarifications, LLMs are limited to input prompts, which may not always include all relevant information. This underscores the need for human review when using AI-generated requirements.

Completeness of Requirements

The results suggest human participants scored higher on completeness for their requirements compared to LLM-generated requirements. This finding indicates that human subjects were better at capturing and including both functional and non-functional requirements in system specifications. The requirements generated by LLMs, although including main system functions, were found to sometimes neglect other important details such as security constraints, performance requirements and exception handling. This could be due to the lack of domain-specific knowledge and the nature of the data that LLMs are trained on (Nijkamp et al., 2023). Incomplete requirements are especially problematic in software engineering, as they can result in system bugs and higher rework in subsequent phases of the development process. This suggests the importance of formal validation and verification processes when working with requirements generated by AI.

Benefits of LLM-Generated Requirements

However, we identified some strengths of LLM-generated requirements in spite of the limitations. For example, it was observed that LLM-generated requirements were highly consistent in their format and composition. The requirements were typically well structured and had consistent sentence structures, which may support ease of understanding and documentation practices.

Moreover, the speed and ability to generate a massive number of requirements in a short period is another strength of LLMs. This feature can be valuable in Agile development practices, where frequent iterations and constant requirement changes are expected (Ahmad et al., 2022). These capabilities indicate that LLMs can be assistive tools in requirement engineering, especially in the early stages of requirement elicitation and specification.

Related Work

Our research aligns with recent studies on applying AI in software engineering. Pearce et al. (2025) found AI-generated solutions can be very productive but not necessarily correct or reliable. Cheng et al. (2024) also highlighted the need for further empirical studies to validate the existing uses of generative AI in requirement engineering.

The differences in performance between human and LLM-generated requirements also support the findings of Khan et al. (2025), which argue that LLMs are not yet ready to completely replace human involvement in requirement engineering tasks. Rather, they are best used to complement human skills.

The fact that these results are consistent with those of other studies suggests the validity of the current research, and the need for further research into the use of AI in requirement engineering.

Insight into AI-Augmented Requirement Engineering

The findings of this study indicate that a combination of human expertise and AI support may be the best approach to requirement engineering. Here, LLMs can be used to draft the initial versions of requirements, and these drafts can be refined by human experts.

This hybrid approach can harness the benefits of both approaches in which the speed and consistency of LLMs can be complemented by the contextual awareness and reasoning skills of human analysts. This approach could potentially result in higher quality requirements, while also reducing the time and effort needed for elicitation and documentation. Moreover, the results underline the need for formalized evaluation processes for AI-generated requirements. Consistent measures and verification processes are needed to ensure the effectiveness of AI-based tools used in software development.

Observations in the Context of the Limitations

The findings on the differences between human and LLM-generated requirements must be viewed in the context of several limitations of this study. The effectiveness of LLM-generated requirements is dependent on the prompts used, which may not adequately represent system requirements. The assessment is also subjective, as it is based on human judgement.

The scenarios and participant size may also impact the results of the study. Differences in domain complexity and participant knowledge may impact the findings. These considerations indicate the need for additional research with larger sample sizes and a range of application domains.

Emerging Research Directions

Our research identifies a number of emerging research opportunities in AI-driven requirement engineering. Advancements in prompt engineering methods may yield better quality and more complete requirements from LLM. Also, incorporating domain expertise into language models could alleviate ambiguity and enhance contextual understanding. Future research could also consider the development of automated evaluation tools that use machine learning to evaluate the quality of requirements. These tools could be used in conjunction with human evaluation to offer scalable requirement analysis.

Another potential avenue is the use of hybrid human-AI systems that adaptively combine human and AI contributions depending on the complexity of the task.

Threats to Validity

The validity of the results discussed in this study may be affected by factors that are well known in empirical software engineering studies. These are associated with internal, external, construct, and conclusion validity, and may influence how the results are interpreted and applied.

The **internal validity** of the study may be influenced by differences in the skill and quality of performance of human participants in the generation of requirements. Variations in knowledge, skills, and experience of the participants, and their understanding of requirement engineering processes may have affected the generated requirements. Further, the Large Language Model's (LLM) responses may have been affected by prompt engineering. While the same prompts and description of each system were used for both human- and LLM-generated requirements, the responsiveness of LLM to the structure of prompts may have led to variability in the outputs that was difficult to control.

The **external validity** of this study might also be limited. The study was based on a small set of software system scenarios, which may not capture the full range of variability, size, and complexity of software systems. Also, the users who generated human requirements may not be representative of the skills and experience of industrial software engineers. This could limit the applicability of the results to large-scale software systems or expert domains.

The interpretation of evaluation metrics in this study may impact on **construct validity**. The key metrics for the analysis of requirements, namely quality, ambiguity and completeness, were identified as key dimensions for evaluation in the literature. But the measurement of these metrics is subject to some evaluator interpretation, especially in detecting ambiguity or establishing completeness. While an evaluation rubric was used to standardize the process, differences in the evaluators' judgement may impact the measurements.

The simplicity of the statistical methods used in the study could impact on the **validity of the conclusions**. Relying on comparisons of means offers straightforward and comprehendible insights but may overlook more intricate statistical associations. Moreover, the limited number of requirement sets may not provide sufficient evidence to support generalized conclusions. However, to improve the validity of the study, consistent evaluation methods and controlled experiment setups were maintained.

Conclusion and Future Work

This study has offered a systematic comparison of human-generated and Large Language Model (LLM)-generated software requirements, focusing on the quality, ambiguity and completeness of the requirements. The use of a controlled experimental approach enabled a direct comparison between the requirements generated by humans and the LLM, as they were both generated under the same conditions. The results suggest that human-generated requirements excel in all comparative aspects. The superior quality and completeness scores indicate

humans better capture the system requirements and do so with greater accuracy and completeness. The lack of ambiguity observed in human-generated requirements also suggests that they are better at communicating requirements concisely and unambiguously.

On the other hand, LLM-generated requirements were found to be consistent in structure and fluent in language. But they also exhibited weaknesses in ambiguity and completeness, with some requirement statements being too vague or missing key aspects of the system. These insights indicate that although LLMs can enhance the requirement generation process, their generated requirements may need to be validated and refined by humans to achieve the level of quality required for software engineering projects. This research showcases the opportunities of combining AI-driven tools with human involvement in requirement engineering processes. A combination of using LLMs to generate initial drafts and human analysts to validate and refine the requirements could be a promising solution. This may lead to improved efficiency without compromising software requirements.

Future studies may increase the number of system scenarios and participants to enhance the generalizability of the results. The use of sophisticated statistical methods may also yield more insights into the differences in performance between human and AI-generated requirements. Further, the use of domain-specific information in LLMs and advances in prompt engineering may lead to more accurate and complete requirements. The use of automated assessment tools may also lead to the creation of more efficient methods of evaluating quality in AI-supported systems.

References

Ahmad, K., Abdelrazek, M., Arora, C., Bano, M., & Grundy, J. (2022). Requirements Engineering for Artificial Intelligence Systems: A Systematic Mapping Study. *ArXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2212.10693

Arora, C., Grundy, J., & Abdelrazek, M. (2023, November 1). *Advancing Requirements Engineering through Generative AI: Assessing the Role of LLMs*. ArXiv.org. https://doi.org/10.48550/arXiv.2310.13976

Boehm, B., & Basili, V. R. (2001). Software Defect Reduction Top 10 List. *Computer*, *34*(1), 135–137. https://doi.org/10.1109/2.962984

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., Bohg, J., Demszky, D., & Doumbouya, M. (2021). On the Opportunities and Risks of Foundation Models. *ArXiv:2108.07258 [Cs]*, *1*(1). https://arxiv.org/abs/2108.07258

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., & Hesse, C. (2020). Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems*, *33*, 1877–1901. https://proceedings.neurips.cc/paper_files/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html?utm_source=transaction&utm_medium=email&utm_campaign=linkedin_newsletter

Cheng, H., Husen, J. H., Lu, Y., Teeradaj Racharak, Yoshioka, N., Naoyasu Ubayashi, & Hironori Washizaki. (2024). Generative AI for Requirements Engineering: A Systematic Literature Review. *Software Practice and Experience*. https://doi.org/10.1002/spe.70029

Fan, A., Beliz Gokkaya, Harman, M., Mitya Lyubarskiy, Sengupta, S., Yoo, S., & Zhang, J. M. (2023). Large Language Models for Software Engineering: Survey and Open Problems. *ArXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2310.03533

Filippi, S., & Motyl, B. (2024). Large Language Models (LLMs) in Engineering Education: A Systematic Review and Suggestions for Practical Adoption. *Information*, *15*(6), 345–345. https://doi.org/10.3390/info15060345

Hemmat, A., Sharbaf, M., Kolahdouz-Rahimi, S., Lano, K., & Tehrani, S. Y. (2025). Research directions for using LLM in software requirement engineering: a systematic review. *Frontiers in Computer Science*, *7*. https://doi.org/10.3389/fcomp.2025.1519437

Kampelopoulos, D., Tsanousa, A., Vrochidis, S., & Kompatsiaris, I. (2025). A review of LLMs and their applications in the architecture, engineering and construction industry. *Artificial Intelligence Review*, *58*(8). https://doi.org/10.1007/s10462-025-11241-7

Khan, J. A., Qayyum, S., & Dar, H. S. (2025). Large Language Model for Requirements Engineering: A Systematic Literature Review. *Research Square (Research Square)*. https://doi.org/10.21203/rs.3.rs-5589929/v1

McIntosh, T. R., Susnjak, T., Arachchilage, N., Liu, T., Xu, D., Watters, P., & Halgamuge, M. N. (2025). Inadequacies of Large Language Model Benchmarks in the Era of Generative Artificial Intelligence. *IEEE Transactions on Artificial Intelligence*, 1–18. https://doi.org/10.1109/tai.2025.3569516

Nijkamp, E., Pang, B., Hayashi, H., Tu, L., Wang, H., Zhou, Y., Savarese, S., & Xiong, C. (2023). *CodeGen: An Open Large Language Model for Code with Multi-Turn Program Synthesis*. ArXiv.org. https://doi.org/10.48550/arXiv.2203.13474

Pearce, H., Ahmad, B., Tan, B., Dolan-Gavitt, B., & Karri, R. (2025). Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions. *Communications of the ACM*, *68*(2), 96–105. https://doi.org/10.1145/3610721

Sommerville, I. (2016). *Software engineering* (10th ed.). Pearson.

Zadenoori, M. A., Dąbrowski, J., Alhoshan, W., Zhao, L., & Ferrari, A. (2025). *Large Language Models (LLMs) for Requirements Engineering (RE): A Systematic Literature Review*. ArXiv.org. https://arxiv.org/abs/2509.11446

Wiegers, K. E., & Beatty, J. (2013). *Software requirements* (3rd ed.). Microsoft Press.