Quality Evaluation of Software Functional Requirements Generated by LLMs: A Systematic Mapping Study

Lucas M. Lourenço[1], **Carla Bezerra**[1], **Allan P. Marques**[1]

[1]Programa de Pós-graduação em Computação (PCOMP)
Universidade Federal do Ceará (UFC)
Caixa Postal 63900-000 – Quixadá – CE – Brasil

`lucasmlourenco.es@gmail.com, carlailane@ufc.br, allanpeixoto38@gmail.com`

Abstract. Research Context: *Large Language Models (LLMs) have been increasingly applied in software engineering, especially in the generation of functional requirements for information systems.* ***Scientific and/or Practical Problem:*** *However, the quality of these requirements still raises concerns, such as ambiguities, incompleteness, and dependency on prompts.* ***Proposed Solution and/or Analysis:*** *This study conducts a systematic mapping study to analyze how the literature has evaluated requirements generated by LLMs.* ***Related IS Theory:*** *The work is aligned with the sociotechnical perspective, considering information systems requirements as both technical and social artifacts.* ***Research Method:*** *A systematic mapping study was conducted, reviewing 1,875 studies and selecting 51 primary studies published between 2020 and 2025.* ***Summary of Results:*** *The findings indicate a diversity of evaluation practices, combining traditional criteria and NLP metrics, with advantages in automation and standardization, but limitations regarding the absence of benchmarks and validation in industrial contexts, especially in the development of information systems.* ***Contributions and Impact to IS area:*** *The study contributes to academia by consolidating evaluation approaches and identifying gaps, and to industry by supporting the understanding of risks and opportunities in the use of LLMs in the requirements engineering of information systems.*

1. Introduction

In recent years, the advancement of Large Language Models (LLMs), such as GPT-3 and GPT-4 [Achiam et al. 2023], has driven research in various areas of computing, including software engineering. These models, capable of processing natural language at scale, have been applied to tasks such as code generation, documentation review, and, more recently, software requirements production [Touvron et al. 2023, Zhao et al. 2023a].

In Requirements Engineering (RE), LLMs have been explored to support elicitation, documentation, and analysis activities, offering gains in productivity, standardization, and reduced human effort [Ferrari et al. 2023]. The automation of requirements writing represents a significant advance, since manual production often suffers from problems of inconsistency, ambiguity, and lack of completeness [Sommerville 2011, Wiegers and Beatty 2013]. However, the adoption of LLMs in this context still faces important challenges, such as hallucinations [Ji et al. 2023], excessive dependence on prompts, lack of domain knowledge, and difficulty in ensuring semantic correctness and alignment with stakeholders' needs.

Functional requirements (FRs), in particular, play a central role in this debate, as they explicitly describe what the system must accomplish in terms of observable behavior. The literature highlights that failures in FRs directly impact subsequent stages of the software lifecycle — design, implementation, testing, and maintenance — and are among the main causes of project failure [Pohl 2010, Fernández et al. 2017]. Thus, when LLMs are used to generate FRs, it becomes essential to understand whether these artifacts possess quality attributes such as clarity, completeness, consistency, correctness, and testability [ISO 2018, Davis et al. 1993].

The evaluation of the quality of these requirements is another critical point. Studies have employed both traditional attributes — clarity, completeness, and consistency (IEEE, 2017) — and automatic NLP metrics, such as BLEU, ROUGE, and BERTScore [Papineni et al. 2002, Lin 2004, Zhang et al. 2020]. This diversity reflects the exploratory nature of the field but highlights the lack of standardization, hindering comparisons and the consolidation of best practices. Furthermore, the statistical nature of LLMs can produce texts that are linguistically correct but semantically inaccurate, reinforcing the need for hybrid evaluation mechanisms that combine automatic metrics with human validation [Hemmat et al. 2025].

This scenario resonates with the strategic research challenges in Information Systems in Brazil, especially Challenge 4 – Sociotechnical Vision of Information Systems from GranDSI-BR [Araújo and Suzana 2017], which emphasizes the importance of considering systems as complex sociotechnical arrangements. Assessing the quality of requirements generated by LLMs involves not only technical aspects but also their interpretation, validation, and use by stakeholders, contributing to a critical sociotechnical view of the impacts of this technology.

Therefore, this study conducts a systematic mapping study of the literature to investigate how the quality of functional requirements generated by LLMs has been evaluated. Searches were carried out in well-established computing databases, resulting in the selection of 51 primary studies. The results, organized around research questions, address methods, metrics, advantages, limitations, and proposed approaches, offering a comprehensive view of the topic and contributing both to professional practice and to scientific advancement.

2. Quality of Functional Requirements

The evaluation of the quality of functional requirements (FRs) is an essential pillar of Requirements Engineering, as these artifacts describe what the system must accomplish in terms of observable behavior. The quality of FRs directly impacts subsequent activities such as design, implementation, testing, and validation, influencing communication among stakeholders and the accuracy of development [Sommerville 2011, Wiegers and Beatty 2013].

Standards and best practices, such as [Committee and Board 1998] and [ISO 2018], establish quality attributes that must be observed. Among them is clarity, which refers to ease of understanding, avoiding vague or subjective terms such as "fast" or "intuitive" [Committee and Board 1998, Sommerville 2011]. Clarity is directly related to non-ambiguity, as a requirement is only considered of good quality if it is interpreted in the same way by different readers [Wiegers and Beatty 2013].

Another central attribute is completeness, which requires that the requirement address all relevant conditions, including main flows, exceptions, and constraints. The literature emphasizes that incomplete requirements are among the most frequent causes of system failures [Pohl 2010, Fernández et al. 2017]. Complementarily, consistency ensures that there are no contradictions among different FRs, avoiding conflicts of rules and objectives [ISO 2018, Davis et al. 1993].

Correctness is also essential, ensuring that requirements adequately represent stakeholders' needs and business goals. [Wiegers and Beatty 2013] emphasize that a requirement may be well written, but if it does not correctly reflect the client's intent, it compromises the entire system. This concern appears in empirical studies that highlight recurring gaps between the formulation of FRs and the actual expectations of users [Fernández et al. 2017, Wagner et al. 2019].

Testability, or verifiability, is another fundamental attribute, defined by the possibility of deriving clear and objective acceptance criteria from the requirement. According to [ISO 2018], an FR must allow its implementation to be unequivocally verified, usually through test cases or formal inspections.

Finally, standardization is highlighted as a mechanism to unify the language and structure of FRs, increasing readability and reducing ambiguities. The standard [ISO 2018] recommends the use of templates and controlled vocabularies, a practice also reinforced by [Wiegers and Beatty 2013]. In addition, the absence of redundancy and adequacy ensure that FRs do not repeat unnecessary information and that each requirement effectively contributes to the system's objectives.

In recent years, the topic has received renewed attention in the context of applying Natural Language Processing (NLP) techniques and, more recently, large language models (LLMs). The study by [Zhao et al. 2021] shows that automatic methods can support the detection of problems in requirements, especially in attributes such as clarity and consistency. Systematic reviews have also explored the state of the art in automatic requirements evaluation, highlighting metrics and tools aimed at supporting quality analysis [Umar and Lano 2024].

Thus, the consolidation of these attributes into a quality model provides a solid basis for evaluating functional requirements, whether written manually or generated by automatic tools. At the same time, it becomes evident that evaluation must be both structural (focused on linguistic and formal attributes) and semantic (focused on correctness and business adequacy), especially when applied to FRs produced by LLMs.

3. Related Work

The study by [Hou et al. 2024] provides a comprehensive overview of the role of LLMs in Software Engineering by gathering 395 works published between 2017 and 2024. Among the main aspects analyzed are the most widely used models, optimization strategies, evaluation methods, and engineering tasks in which such models have been applied, including Requirements Engineering. Despite its panoramic nature, the investigation does not delve into the analysis of the quality of functional requirements generated by LLMs, an issue that constitutes the core of the present research.

Taking a more specific focus, [Huang et al. 2025] systematize how prompt engineering techniques have been applied in Requirements Engineering activities. The work proposes a taxonomy that relates prompting patterns — such as few-shot, chain-of-thought, and retrieval-augmented generation — to different tasks, ranging from elicitation to validation and traceability. Although the discussion reveals how interaction choices can impact the outcomes of language models, the focus is not on evaluating the quality of the generated requirements but rather on strategies for prompt formulation.

More classical discussions on requirements quality appear in the work of [Umar and Lano 2024], which compiles and organizes models and frameworks aimed at defining and evaluating attributes such as clarity, consistency, completeness, and testability. The contribution of this study lies in consolidating conceptual foundations that still serve as a reference for evaluating software requirements, functioning as an essential theoretical basis for research that, like the present one, investigates the quality of requirements generated by LLMs.

The research by [Siddeshwar et al. 2024] broadens the discussion by compiling 122 articles on AI frameworks applied to requirements elicitation. This review explores the activities that benefit from machine learning and NLP techniques, as well as the types of data and reported performance levels. The conclusions point to persistent challenges, such as ambiguity and the dynamic nature of stakeholders' needs, but do not address the evaluation of artifacts generated by LLMs, highlighting the relevance of studies more directly focused on the quality of these requirements.

Meanwhile, [Santos et al. 2025] focus on the automatic generation of user stories, identifying available corpora, employed techniques, and different forms of evaluation. The authors observe a lack of standardized guidelines and emphasize that the field is still at an early stage of maturity. Although the focus is on user stories rather than traditional functional requirements, the study contributes by highlighting the same methodological gap faced by research on AI-generated requirements: the absence of clear criteria and robust benchmarks.

Unlike previous reviews, which range from broad overviews to discussions on prompting, present foundations of requirements quality, or analyze applications in elicitation and user story generation, this work focuses directly on evaluating the quality of functional requirements generated by LLMs.

4. Methodology

The objective of this work is to identify, analyze, and synthesize the methods, metrics, and criteria used in evaluating the quality of functional software requirements generated by LLMs, with the aim of understanding the advances, challenges, and gaps present in the literature. The construction of the research protocol was based on the guidelines proposed by [Kitchenham and Charters 2007], encompassing the phases of planning, execution, and results analysis.

4.1. Research Questions

Based on the defined objective, the following Research Questions (RQs) were formulated:

RQ1: Which LLM models have been used in the generation of functional software requirements, and which ones stand out? **Objective:** Identify the LLM models used in generating functional requirements and highlight those with the best performance reported in the literature.

RQ2: From a quality evaluation perspective, what are the main advantages and limitations of LLMs in generating functional software requirements, according to the literature? **Objective:** Analyze the main advantages and limitations of LLMs in generating functional requirements, highlighting benefits and challenges reported in the literature to guide future research and applications.

RQ3: What has been the quality of the requirements generated by LLMs? **Objective:** Assess the current state of the quality of functional requirements produced by LLMs, discussing their variability, strengths, and weaknesses in order to understand the reliability of these artifacts in different usage contexts.

RQ4: What are the criteria and metrics used to evaluate the quality of functional software requirements generated by LLMs? **Objective:** Map and systematize the evaluation criteria and metrics used in the literature, aiming to identify patterns, gaps, and levels of standardization that can guide future research and practices in quality assessment.

RQ5: What approaches have been proposed to validate or improve the quality of functional requirements generated by LLMs? **Objective:** Investigate the strategies developed in the literature to evaluate or enhance the quality of generated requirements, identifying best practices and opportunities for methodological consolidation.

RQ6: What research gaps still persist in the field of evaluating functional requirements generated by LLMs? **Objective:** Highlight the aspects not yet explored or insufficiently consolidated in the literature, such as the absence of standardized metrics, low generalization, and methodological challenges, in order to guide future research and strengthen the field.

4.2. Research Protocol Construction

The construction of the research protocol followed the Population, Intervention, Comparison, Outcome, Context (PICOC) strategy (Table 1) [Liberati et al. 2009], which guided the definition of the essential elements of the review. To ensure the comprehensiveness of the search, five databases widely used in the computing field were selected: ACM Digital Library, IEEE Digital Library, Science@Direct, Scopus, and Springer Link. The considered period included studies published between January 2020 and June 2025, aligning with the recent growth of LLM applications in requirements engineering. The stages of the review are presented in Figure 1.

Two search strings were used, one for the Science@Direct database, which has a restriction on the number of connectors:

(“requirements engineering" OR “software requirements" OR “requirements evaluation" OR “requirements validation”) AND (“LLMs" OR “Large Language Models" OR “language models" OR “generative AI" OR “generative artificial intelligence")

And another for the other databases:

(“requirements engineering" OR “software requirements" OR “functional

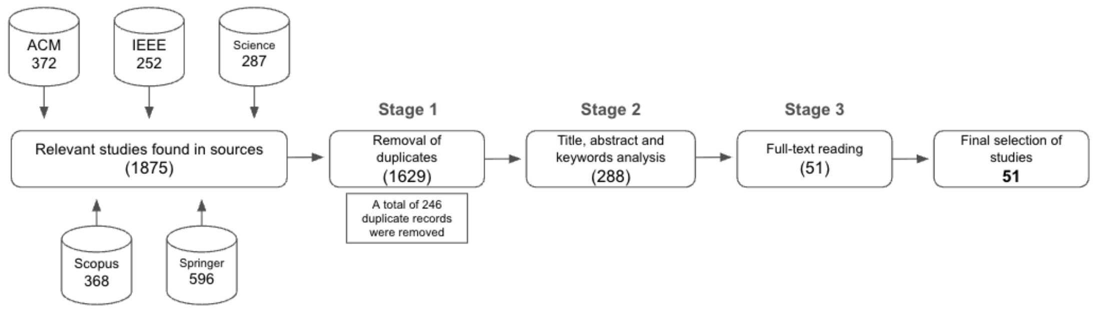
ACM
372
IEEE
252
Science
287
Relevant studies found in sources
(1875)
Scopus
368
Springer
596
Stage 1
Removal of duplicates
(1629)
A total of 246 duplicate records were removed
Stage 2
Title, abstract and keywords analysis
(288)
Stage 3
Full-text reading
(51)
Final selection of studies
51

Figure 1. Search and selection review process.

requirements" OR "requirements validation" OR "requirements evaluation") AND ("LLMs" OR "Large Language Models" OR "language models" OR "generative AI" OR "generative artificial intelligence")

Table 1. PICOC Protocol used in the SML

PICOC Protocol	
Population	Software requirements, requirements artifacts, requirements specification
Intervention	Large Language Models (LLMs), generative AI
Comparison	Not applicable
Outcomes	Functional requirements quality, quality assessment, quality metrics, quality criteria, requirements validation
Context	Software functional requirements validation

4.3. Definition of Inclusion and Exclusion Criteria

The online tool Parsifal[1] was used to automate part of the selection process, facilitating the identification and elimination of duplicate studies.

To ensure the relevance and quality of the studies included in the mapping, specific inclusion and exclusion criteria were established (Table 2). These criteria aim to guarantee that only truly pertinent works are considered to answer the research questions.

Table 2. Inclusion and exclusion criteria used in the selection of studies

ID	**Inclusion criteria**	**ID**	**Exclusion criteria**
I1	Articles in English	E1	Duplicate articles
I2	Articles published from 2020 onwards	E2	Articles published before 2020
I3	Primary studies	E3	Non-peer-reviewed studies (book chapters, technical reports, documents, abstracts, or presentations)
I4	Studies addressing the validation of functional requirements generated by LLMs	E4	Studies not addressing the evaluation of functional requirements generated by LLMs
		E5	Secondary or tertiary studies (such as systematic reviews or mappings)
		E6	Grey literature

4.4. Study Selection

The study selection process involved three reviewers. Two researchers independently conducted the screening and selection of studies based on the defined inclusion and exclusion criteria. Subsequently, a third reviewer, a specialist in software requirements with

[1]https://parsif.al

more than ten years of experience, performed a final validation of the selected studies. Disagreements were discussed until consensus was reached, ensuring consistency and reducing individual bias.

The study selection was carried out in four stages (Table 3). Initially, 1,875 studies were imported from the databases. Then, in the duplicate removal stage, this number was reduced to 1,629 studies. In the third stage, the inclusion and exclusion criteria (Table 2) were applied, resulting in 288 selected studies. Finally, after full-text reading, 51 studies remained and were included in the mapping. All selected studies, along with additional details about each of them, are available at `https://zenodo.org/records/18558380`.

Table 3. Search results by database

Database	Found	Duplicates	Rejected	Accepted
ACM Digital Library	372	19	346	7
IEEE Digital Library	252	51	181	20
Science@Direct	287	11	274	2
Scopus	368	85	260	21
Springer Link	596	80	514	1
Total	**1875**	**246**	**1575**	**51**

5. Results

The application of the search and selection protocol resulted in a set of 51 primary studies that meet the defined inclusion criteria. This section presents the main findings from the analysis of these works, focusing on the methods, metrics, and criteria employed in evaluating the quality of functional software requirements generated by LLMs. In addition, the proposed approaches, the models used, and the reported limitations are discussed, providing a comprehensive view of the current state of research in the field. The results of the extraction and analysis of the selected studies are available at `https://zenodo.org/records/18558380`.

Figure 2 presents the number of articles (out of the 51 accepted) per year. The figure shows a notable increase in related studies since 2022, with a considerable rise in 2023 and 2024, which indicates that researchers have shown growing interest in this topic.

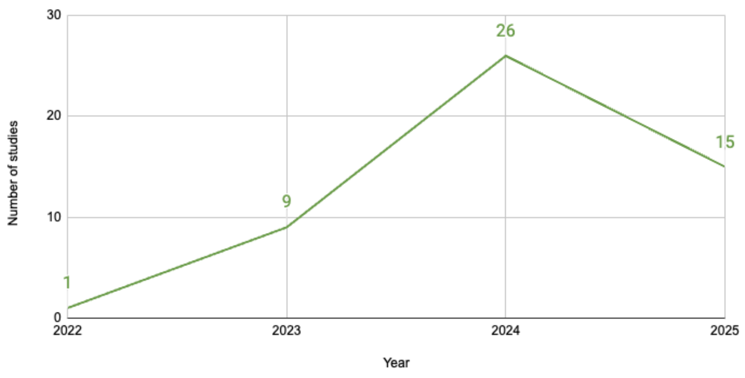

Figure 2. Accepted studies per year.

5.1. RQ1: LLMs used in the generation of functional software requirements

The analyzed studies indicate a very clear predominance of the GPT family (OpenAI), which stands out as by far the most widely used and also the most prominent in the generation of functional requirements. This dominance suggests that, in the current scenario, OpenAI's models serve as a practical and comparative reference in relation to other approaches.

In addition, the presence of other LLM families is observed, albeit on a smaller scale, but they contribute to diversifying the experiments and analyses carried out in the studies. Table 4 presents a synthesis of the identified tools and the respective studies that mention them.

Table 4. LLMs used for generating functional software requirements according to the literature

Provider	Family	Models (cited versions)	Studies (codes)
OpenAI	GPT	GPT-2, GPT-3, GPT-3.5, GPT-4, GPT	S1, S2, S3, S4, S5, S6, S7, S8, S9, S10, S11, S12, S13, S14, S15, S16, S17, S18, S19, S20, S21, S22, S23, S24, S25, S26, S27, S28, S29, S30, S31, S32, S33, S34, S35, S36, S37, S38, S39, S40, S41, S42, S43, S44, S45, S46, S47, S48, S49, S50, S51
Meta (Facebook)	LLaMA	LLaMA-2, LLaMA-3, LLaMA	S5, S6, S12, S19, S20, S21, S25, S40, S44, S47, S49, S50
Google	PaLM	PaLM, PaLM-2	S23, S40, S43, S47, S48, S51
Google	T5	T5	S9, S13, S28
Google	Gemini	Gemini	S23, S40, S43, S47
Google Research	Pegasus	Pegasus	S20
Anthropic	Claude	Claude, Claude 2	S11, S16, S21, S51
Microsoft	Bing Chat	Bing Chat	S1, S28
Open-source	Falcon	Falcon	S19, S30
Open-source	Mistral	Mistral	S8, S19
Others	BERT-family	BERT	S7, S18, S22, S27, S33
Others	Not specified	Other/Not specified	S10, S29, S39, S41

The literature highlights the predominance of OpenAI's GPT family models, especially GPT-4, as the central reference in the generation of functional software requirements. These models stand out for their superior performance in quantitative metrics (precision, recall, F1-score, BLEU, METEOR, ROUGE) and for greater consistency compared to human expert evaluations, in addition to showing solid results in tasks such as requirements translation, SRS generation, automatic validation, and inconsistency correction.

Their wide availability, consolidated documentation, and mature support largely explain this predominance. On the other hand, alternative models such as LLaMA, Gemini, and Claude appear in specific contexts, signaling the scientific community's pursuit of diversification, although there is still a strong concentration on the use of OpenAI's LLMs.

Finally, it is important to note that not all studies conducted direct comparisons between different LLMs, which constitutes a relevant gap in the literature and limits the possibility of drawing more robust conclusions about the relative performance between model families.

5.2. RQ2: Main advantages and limitations of LLMs in generating functional software requirements

The analysis of the selected studies (Table 5 and Table 6) shows that the use of LLMs in generating functional software requirements combines significant advances with relevant challenges, shaping a field that is still consolidating.

Among the most notable advantages, automation of the process and the consequent productivity gains are mentioned in [Ramasamy et al. 2024, Nakagawa and Honiden 2023, Vogelsang 2024]. The reduction of ambiguities and textual redundancies was highlighted in [Voria et al. 2025, Dearstyne et al. 2024, Blasek et al. 2023], while the ability to generate complementary artifacts — such as test cases, acceptance criteria, and complete specifications (SRS) — appears in [Liu et al. 2024, Lubos et al. 2024, Bajaj et al. 2022]. Standardization and textual clarity, factors that contribute to communication among stakeholders and traceability, are emphasized in [Schwedt and Ströder 2025, Norheim and Rebentisch 2024, Sabetzadeh and Arora 2025].

On the other hand, the studies also report several limitations. Restricted generalization to specific contexts was observed in [Sharma et al. 2024, Ronanki et al. 2023], while low accuracy in specialized domains was discussed in [Ruan et al. 2023, Binder and Mezhuyev 2024]. The occurrence of internal inconsistencies among generated requirements is reported in [Bertram et al. 2023] and [Sami et al. 2024]. Issues of incompleteness and persistent ambiguity are recurrent in [Santos et al. 2024, Krishna et al. 2024, Zhao et al. 2023b], and cases of hallucinated content were reported in [Ferrari and Spoletini 2025, Nicklas and Mozelius 2024]. Concerns about bias in training data [Zitouni et al. 2025, Arora et al. 2024] and high computational costs [Xu et al. 2024, Shah et al. 2025] are also recurring, alongside issues of context limits [Jain et al. 2023] and privacy and security [Wang et al. 2024, Zhang et al. 2024].

Thus, the adoption of LLMs in Requirements Engineering must be understood through a dialectic between benefits and weaknesses. On one hand, the gains in efficiency, clarity, and standardization are evident; on the other, the risks of errors, incompleteness, and biases demand well-structured mitigation strategies. Ultimately, the results suggest that LLMs should not be seen as substitutes for traditional practices, but as complementary tools, whose effectiveness depends on integration with human validation, robust methodologies, and technical instruments capable of reducing uncertainties and increasing the reliability of generated requirements.

5.3. RQ3: Quality of requirements generated by LLMs

The quality of functional requirements generated by LLMs has been considered promising, although variable, depending on the context, the application domain, the technique employed—such as prompt engineering or fine-tuning—and the level of human intervention.

Positive aspects include completeness and clarity, since the requirements are often understandable, well-structured, and frequently comparable to those written by humans. Efficiency is also highlighted, as LLMs can generate requirements in significantly less time than traditional methods. Another important point is measurable quality, with studies reporting high metrics, such as F1-scores above 90%, as well as strong performance

Table 5. Advantages of LLMs in requirements generation and studies that mention them

Advantage	Studies (codes)
Automation and increased efficiency/productivity	S1, S2, S3, S5, S6, S8, S9, S10, S11, S12, S13, S15, S16, S17, S18, S19, S20, S21, S22, S23, S24, S25, S27, S28, S29, S30, S31, S32, S33, S34, S35, S36, S37, S38, S39, S40, S41, S42, S45, S47, S48, S49, S50, S51
Generation of additional artifacts (acceptance criteria, test cases, diagrams)	S1, S4, S5, S7, S8, S10, S11, S13, S15, S17, S19, S21, S22, S24, S25, S26, S28, S30, S31, S35, S36, S37, S39, S41, S42, S43, S44, S46
Improved accuracy, coherence, and consistency	S3, S4, S8, S9, S12, S14, S16, S17, S18, S19, S20, S23, S25, S27, S28, S29, S34, S35, S36, S37, S39, S40, S41, S42, S44, S48, S49, S51
Textual clarity, standardization, and editability	S2, S3, S4, S5, S9, S10, S12, S13, S15, S16, S19, S20, S21, S22, S23, S25, S27, S29, S35, S37, S39, S40, S42, S46, S47, S48, S49, S51
Performance comparable to humans	S2, S3, S4, S5, S9, S12, S15, S16, S17, S19, S21, S23, S24, S26, S27, S29, S33, S37, S39, S42, S48
Adaptation to domains and scalability	S5, S7, S8, S9, S11, S15, S19, S23, S25, S26, S30, S31, S33, S35, S36, S39, S40, S43, S45
Completeness and coverage (edge cases, gaps)	S1, S4, S7, S12, S16, S19, S21, S22, S24, S27, S29, S33, S38, S41, S42, S44, S51
Reduction of ambiguities and redundancies	S2, S3, S9, S10, S19, S21, S31, S39, S47, S48
Support for ideation and human–LLM collaboration	S3, S14, S21, S22, S32, S34
Economic feasibility (low cost per execution/artifact)	S6, S7, S14, S31, S36
Privacy/security considerations	S2, S5, S11

in precision, recall, and BLEU in extraction and generation tasks. Their innovative capacity is also noteworthy, as models can suggest new requirements or requirements not previously considered by stakeholders. In addition, successful applications have been identified in domains such as user stories, acceptance criteria, and formal specifications, where LLMs have demonstrated competitive or even superior performance compared to manual approaches and traditional tools.

Persistent challenges, on the other hand, include variability in quality, which can result in ambiguous, irrelevant, or technically incomplete requirements. Prompt sensitivity is another critical issue, since small changes in input may lead to considerably different outputs. Human supervision also remains essential, as review, refinement, and validation are necessary to ensure accuracy, completeness, and alignment with the system scope. Finally, without adaptation or specific instructions, LLMs show limitations in domain and contextual knowledge, which undermines the correct capture of complex relationships, constraints, or specialized terminology.

In summary, requirements generated by LLMs present good initial quality and can serve as drafts or support for elicitation. However, they are still not suitable for direct use without expert review, especially in critical or highly complex systems. The literature highlights relevant advances but also reinforces the need for technical and methodological improvements to ensure greater reliability and contextual adequacy.

5.4. RQ4: Criteria and metrics used to assess the quality of functional software requirements generated by LLMs

The analyzed studies indicate a wide variety of criteria and metrics for assessing the quality of functional software requirements generated by LLMs. Regarding the criteria, traditional aspects of requirements engineering are observed, such as accuracy, correctness, completeness, and clarity, as well as elements related to readability, standardization, absence of ambiguity, and redundancy. In some cases, specific normative criteria, such as

Table 6. Limitations of LLMs in generating functional software requirements according to the literature

Limitation	Studies (codes)
Limited generalization / domain dependence	S2, S3, S10, S11, S12, S13, S14, S15, S17, S18, S19, S20, S21, S22, S23, S24, S25, S26, S27, S28, S29, S30, S31, S32, S33, S34, S35, S36, S37, S38, S39, S40, S41, S42
Low accuracy / errors	S2, S3, S10, S11, S12, S13, S14, S17, S18, S19, S20, S21, S22, S23, S24, S25, S26, S27, S28, S29, S30, S31, S32, S33, S34, S35, S36, S37, S51
Dependence on data / issues / experts	S2, S3, S14, S15, S18, S20, S23, S24, S25, S27, S28, S29, S30, S31, S33, S35, S36, S37, S38, S39, S40, S41, S42, S43, S44
Inconsistency / limited coherence	S2, S3, S10, S11, S12, S13, S14, S15, S17, S19, S20, S21, S23, S24, S25, S27, S29, S31, S32, S33, S34, S51
Computational cost / resources	S10, S12, S21, S22, S24, S26, S27, S30, S32, S33, S36, S38, S39, S40, S41, S42, S43
Incompleteness / lack of coverage	S2, S3, S10, S12, S13, S14, S17, S19, S20, S22, S23, S25, S27, S29, S31, S32, S33, S51
Context limit	S10, S12, S16, S17, S19, S23, S24, S27, S34, S35, S36, S37, S39, S40
Hallucinations / Fabricated information	S2, S12, S14, S17, S19, S21, S27, S34, S35, S36, S38, S39, S40, S51
Bias in results	S3, S12, S14, S21, S23, S25, S28, S31, S35, S36, S37, S39, S40
Ambiguity / lack of clarity	S19, S21, S22, S23, S25, S27, S29, S37, S42, S44, S45, S47
Privacy / Security	S2, S3, S5, S6, S21, S23, S27, S30, S31, S36, S47
Low interpretability	S2, S9, S17, S21, S23, S26, S27, S29, S33, S42

those defined by ISO 26262, were also considered. Table 7 presents the main identified criteria, along with the studies that mention them.

Table 7. Evaluation criteria and studies that mention them

Criterion	Studies (codes)
Accuracy	S3, S6, S8, S9, S11, S12, S16, S19, S20, S21, S23, S25, S26, S29, S31, S39, S40, S41, S42, S43, S44, S45, S48, S49, S51
Correctness	S4, S5, S11, S12, S13, S19, S20, S21, S24, S27, S28, S30, S33, S38, S45, S50, S51
Completeness	S1, S4, S5, S7, S12, S15, S19, S21, S22, S23, S30, S33, S37, S45, S48
Adequacy/Relevance	S2, S3, S5, S16, S20, S22, S23, S28, S36, S37, S38, S48, S50
Readability/Understandability	S2, S7, S16, S17, S19, S22, S23, S27, S30, S37, S50
Standardization/Compliance	S2, S12, S20, S21, S24, S25, S27, S30, S31, S39, S41
Unambiguity	S2, S9, S19, S21, S22, S27, S29, S31, S48, S50
Absence of redundancy	S4, S5, S19, S28, S31, S47, S48
Clarity	S12, S19, S21, S23, S32, S46, S48
Coherence	S2, S13, S16, S44
Validity/Validation	S13, S24, S47
ISO 26262 (quality crit.)	S1

In addition to the criteria, different metrics were employed to quantify the quality of the requirements. Among them are metrics derived from machine learning and automatic natural language evaluation, such as precision, recall, F1-score, BLEU, ROUGE, METEOR, BERTScore, and perplexity. More general metrics were also used, such as accuracy and coverage, as well as direct evaluations conducted with users through questionnaires. Table 8 summarizes these metrics and lists the studies that applied them.

In general, the literature emphasizes criteria such as precision, completeness, and correctness, as well as established automatic NLP metrics such as BLEU and ROUGE. On the other hand, more complex aspects, such as coherence, traceability, and metrics that capture semantic and contextual nuances, appear in a smaller number of studies. This highlights both the relevance of the advances achieved and the need to broaden and standardize evaluation approaches in order to ensure greater reliability and applicability of LLM-generated requirements in industrial contexts.

Table 8. Evaluation metrics and studies that mention them

Metric	Studies (codes)
Precision (ML)	S3, S6, S8, S9, S11, S12, S16, S19, S20, S21, S23, S25, S26, S29, S30, S31, S39, S40, S41, S42, S43, S44, S45, S48, S49, S51
Recall/Sensitivity	S6, S8, S12, S16, S20, S23, S25, S26, S29, S30, S31, S39, S40, S41, S42, S43, S44, S51
BLEU	S9, S10, S12, S15, S16, S29, S36, S39, S42, S51
F1-score	S6, S25, S30, S31, S39, S41, S43, S44, S51
User Evaluation (questionnaire)	S16, S17, S19, S22, S32, S35, S37, S48
ROUGE	S12, S16, S18, S29, S30, S39, S42, S51
Accuracy	S9, S23, S25, S40, S51
Coverage	S2, S13, S16, S44
METEOR	S12, S16, S18
BERTScore	S13, S24, S47
Perplexity	S17, S18

5.5. RQ5: Approaches proposed to validate or improve the quality of functional requirements generated by LLMs

The results of RQ4 reveal a scenario of methodological diversity regarding the strategies adopted to validate or improve the quality of software functional requirements generated by LLMs. The literature focuses on different fronts, ranging from prompt engineering practices to specific frameworks supporting validation. In general, these approaches are not mutually exclusive but rather complementary, aiming to mitigate well-known model limitations such as hallucinations, lack of completeness, or inconsistencies. Table 9 presents a synthesis of the approaches identified in the analyzed studies, highlighting those most frequently mentioned.

Table 9. Approaches proposed to validate or improve the quality of functional requirements generated by LLMs

Approach	Studies (codes)
Prompt Engineering	S1, S2, S3, S4, S5, S6, S11, S12, S13, S14, S15, S17, S18, S19, S20, S21, S22, S23, S24, S25, S26, S27, S28, S29, S30, S31, S32, S33, S34, S35, S36, S37, S38, S39, S40, S41, S42, S43, S44, S45, S46, S47, S48, S49
Semi-automated Processes	S2, S3, S4, S10, S11, S14, S16, S17, S19, S20, S21, S22, S23, S24, S25, S26, S27, S28, S29, S30, S31, S32, S33, S35, S36, S37, S38, S41, S42, S50, S51
General Frameworks/Tools	S3, S4, S10, S11, S12, S13, S17, S18, S20, S21, S22, S23, S24, S25, S26, S27, S28, S30, S31, S32, S33, S34, S35, S36, S38, S39, S49, S51
Data Augmentation	S14, S16, S18, S21, S26, S29, S34, S36, S40, S41, S42, S43, S44
Multifaceted/Hybrid Approach	S9, S11, S14, S16, S28, S32, S34, S37, S40, S44, S46, S47
Transformation into Structured Language	S7, S10, S12, S13, S21, S40, S48, S50
User Evaluation/Validation	S17, S22, S25, S37, S44, S51
Text-based Coding	S15, S18, S31, S39
Data Preprocessing	S7, S16
Other Approaches	S24
RECOVER (tool)	S12

The analysis shows that prompt engineering stands out as the most recurring practice, reinforcing its central role in controlling the quality of LLM outputs in functional requirements. However, there is also a significant presence of semi-automated processes and the use of dedicated frameworks/tools, which highlights the community's pursuit of more structured and systematic solutions. Emerging approaches, such as data augmentation, hybrid strategies, and the development of structured languages, broaden the range of available alternatives. Overall, the set of mapped practices demonstrates a trend toward balancing traditional software engineering methods with LLM-specific techniques, signaling a convergence between technological innovation and methodological rigor in the field of requirements engineering.

5.6. RQ6: Research gaps in the field of evaluating functional requirements generated by LLMs

The results of RQ6 show that relevant gaps still persist in the evaluation of functional requirements generated by LLMs. The most recurrent is the absence of standardized metrics and methods, mentioned in 42 studies (82%), which highlights the lack of robust instruments for semantic and contextual criteria. Another critical point is validation restricted to artificial scenarios, cited in 36 studies (71%), reinforcing the scarcity of practical evidence in real or industrial contexts. Similarly, poorly grounded prompt engineering and fine-tuning were highlighted in 36 studies (71%), while limited generalization was pointed out in 33 studies (65%).

In addition, other gaps emerge, such as dependence on domain-specific knowledge (27 studies, 53%), limited exploration of human–LLM interaction (26 studies, 51%), and underexploration of non-functional requirements (23 studies, 45%). Technical issues such as context window limitations (20 studies, 39%), difficulty handling ambiguous requirements (18 studies, 35%), as well as hallucinations (9 studies, 18%), bias (7 studies, 14%), and variability of responses (2 studies, 4%) were also reported.

In summary, as presented in Table 10, the literature still lacks standardization, practical validation, and greater methodological robustness, while technical barriers remain obstacles to the full adoption of LLMs in Requirements Engineering.

Table 10. Research gaps in the evaluation of functional requirements generated by LLMs

Research Gap	Studies (codes)
Absence of standardized metrics and methods for evaluating functional requirements, especially regarding semantic and contextual criteria	S1, S2, S3, S4, S5, S6, S7, S8, S9, S10, S11, S12, S13, S14, S15, S16, S17, S18, S19, S20, S21, S22, S23, S24, S25, S26, S27, S28, S29, S30, S31, S32, S33, S34, S35, S36, S37, S38, S39, S40, S41, S42
Lack of validation in real or industrial contexts, with predominance of evidence from controlled environments or artificial datasets	S2, S3, S6, S7, S9, S10, S12, S13, S14, S16, S17, S18, S19, S20, S21, S22, S23, S24, S25, S27, S28, S29, S30, S32, S33, S34, S35, S36, S37, S38, S39, S40, S41, S43, S44, S46
Poorly grounded prompt engineering and fine-tuning, based on heuristics	S1, S4, S5, S6, S8, S11, S12, S13, S14, S16, S18, S20, S21, S23, S24, S26, S28, S30, S33, S34, S35, S36, S37, S38, S39, S40, S41, S42, S43, S44, S45, S47, S48, S50
Limited generalization, due to the use of restricted domains and small datasets	S3, S4, S7, S9, S10, S12, S13, S14, S15, S16, S17, S18, S20, S21, S22, S24, S25, S26, S28, S29, S30, S31, S33, S34, S36, S38, S39, S40, S42, S44, S45, S46, S47
Dependence on explicit or domain-specific knowledge	S2, S5, S6, S9, S11, S13, S15, S16, S18, S19, S21, S23, S25, S27, S29, S32, S34, S36, S37, S39, S41, S43, S45, S48, S50
Human–LLM interaction underexplored, especially in iterative refinement cycles and educational contexts	S3, S5, S6, S8, S10, S11, S13, S14, S17, S19, S21, S23, S24, S27, S28, S30, S32, S34, S35, S37, S39, S41, S43, S45, S47, S49
Underexploration of non-functional requirements (performance, security, traceability)	S4, S5, S6, S9, S11, S14, S15, S17, S20, S22, S23, S24, S26, S28, S30, S32, S35, S37, S40, S42, S44, S47, S51
Lack of tools, benchmarks, and public datasets for evaluation	S3, S6, S7, S9, S12, S13, S15, S16, S17, S19, S21, S22, S24, S26, S28, S30, S33, S36, S40, S42, S44, S48
Context window (tokens) as a technical barrier	S4, S6, S8, S10, S11, S13, S15, S18, S20, S22, S24, S26, S28, S30, S32, S34, S36, S39, S42, S45
Difficulty in handling ambiguous or complex requirements	S2, S3, S5, S7, S8, S10, S12, S14, S15, S18, S19, S22, S24, S27, S30, S34, S36, S40
Hallucinations (incorrect or non-existent content)	S3, S6, S9, S11, S15, S18, S22, S28, S33
Bias in generated results	S5, S7, S10, S14, S16, S21, S30
Low logical reasoning	S2, S12, S17
Response variability in repeated executions	S6, S18

6. Discussion

The analysis of the studies reveals that LLMs, especially those from the GPT family, already demonstrate the ability to generate clear, consistent, and complete functional requirements, often at levels comparable to or even superior to those of human experts. This potential makes the models useful as tools to support elicitation and documentation, bringing productivity and quality gains in the early stages of development.

Nevertheless, important limitations remain. The predominance of OpenAI models and the lack of methodological diversity compromise the generalization of the results. Furthermore, although classical criteria such as precision and completeness are used, the absence of standardized metrics and common benchmarks hinders the comparison across studies and the consolidation of best practices.

Another challenge is the gap between academia and industry practice. Most of the studies rely on artificial datasets and controlled scenarios, which restricts external validity. It is still unclear to what extent the requirements generated by LLMs remain effective in real-world contexts, marked by constant changes and organizational constraints. There is also a strong concentration on functional requirements, with little attention given to non-functional ones such as security, performance, and traceability. This gap is critical, as complex systems heavily depend on these aspects to meet stakeholders' expectations and regulatory requirements.

Although techniques such as prompt engineering and semi-automated processes appear as potential solutions, they still lack practical validation and robust theoretical grounding. Thus, three challenges stand out for future research: (i) standardization of evaluation metrics; (ii) validation in industrial environments; and (iii) systematic incorporation of non-functional requirements. These points are essential to consolidate the reliable use of LLMs in requirements engineering.

Beyond these technical issues, it is important to consider that the evaluation of the quality of functional requirements generated by LLMs transcends the boundaries of Software Engineering and directly dialogues with the field of Information Systems. Evaluating such requirements does not merely mean ensuring their correctness, but also understanding how they are interpreted, validated, and used in complex sociotechnical arrangements where people, processes, and technology interact.

In this sense, the quality of requirements directly impacts the reliability, acceptance, and success of corporate systems such as ERPs, CRMs, and analytics platforms, influencing information governance practices and organizational decision-making itself. This perspective aligns with Challenge 4 of GranDSI-BR, which emphasizes the need to understand systems as sociotechnical arrangements and broadens the relevance of this discussion for the Information Systems field.

7. Threats to Validity

The conduction of this systematic mapping study is subject to different threats to validity that may influence both the analysis and the interpretation of the findings. To mitigate potential bias during study screening and data extraction, three reviewers were involved in the process. Two reviewers independently analyzed the studies, while a third reviewer, acting as a domain expert, validated the final selection and resolved disagreements through discussion. Although no formal inter-rater agreement metric was computed, this multi-stage review process aimed to strengthen internal validity.

Even so, the classification of the studies may have been affected by the operational definitions adopted, especially in the understanding of what effectively constitutes the evaluation of the quality of functional software requirements. To reduce this risk, calibration meetings were conducted among the reviewers, and a systematic extraction guided by a predefined protocol was applied, ensuring greater consistency and traceability.

With regard to external validity, the results are not necessarily generalizable to all contexts of LLM application, since the search was limited to specific databases and a defined time frame. Furthermore, it is possible that relevant studies were not retrieved due to terminological variations or inherent limitations of the search strings. This risk was mitigated through the validation and refinement of the search strategy, as well as by complementing it with manual searches in the reference lists of the selected studies.

Another threat concerns construct validity, as the extracted metrics and criteria may not fully represent the evaluation of functional requirements quality. This risk was reduced through the prior definition of analysis categories, systematic data extraction, and peer review.

As discussed by [Verdecchia et al. 2023], threats to validity sections in Software Engineering are often treated superficially, as a simple checklist. To avoid this limitation, this study sought to critically reflect on the constraints faced in relation to its central objective: understanding how the quality of functional software requirements generated by LLMs has been evaluated. Nevertheless, we acknowledge that trade-offs between internal and external validity are inevitable, since greater methodological rigor may reduce the scope of the results, while broader scope may compromise part of the consistency and analytical control.

8. Conclusion and Future Work

This systematic mapping study provided an understanding of how the literature has evaluated the quality of functional software requirements generated by LLMs. The analysis of 51 studies showed that these models are already capable of producing requirements with clarity, completeness, and consistency comparable to, and in some cases superior to, those manually developed. Furthermore, automation, textual standardization, and the generation of complementary artifacts stand out, reinforcing the potential of LLMs as support tools in Requirements Engineering.

However, the results also reveal relevant limitations. Challenges remain related to the dependence on prompting strategies, the occurrence of inconsistencies, hallucinations, and gaps in completeness, as well as the lack of specialized domain knowledge. Another critical issue is the lack of validation in industrial environments, since most studies focus on controlled scenarios and artificial datasets. Added to this is the heterogeneity of metrics and criteria, which hinders direct comparisons and the consolidation of best practices. The need to standardize evaluation instruments that capture semantic and contextual nuances, expand the investigation of non-functional requirements, and promote greater integration between automated validation and human review is emphasized.

By bringing the analysis closer to the Information Systems domain, it becomes evident that the evaluation of requirements quality influences not only technical reliability, but also dimensions such as usability, perceived usefulness, and system acceptance in organizational contexts. These factors, in turn, directly affect organizational impact and the success of deployed information systems. In this sense, future research should consider the integration of consolidated IS theories and models, such as information quality, TAM, and UTAUT, to broaden the understanding of how LLM-generated requirements shape organizational practices and contribute to achieving strategic goals.

Thus, this study not only systematizes the current state of the field but also highlights pathways for researchers and practitioners in Software Engineering and Information Systems to advance toward more rigorous, transparent, and applicable methods. This joint perspective strengthens a conceptual and practical foundation capable of guiding the responsible and reliable use of LLMs in requirements generation within complex sociotechnical contexts, connecting technological innovation with broader organizational and scientific impacts.

Acknowledgments

The authors acknowledge the use of OpenAI's ChatGPT to support the translation of excerpts, text revision, and improvement of academic writing in this work. All responsibility for the content remains solely with the authors.

References

Achiam, O., Adler, S., et al. (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Araújo, R. and Suzana, R. (2017). Grand research challenges in information systems in brazil 2016–2026. *Brazilian Computer Society. Clodis Boscarioli Renata Araujo and Rita Suzana*, 5(1):2016–2026.

Arora, C., Grundy, J., and Abdelrazek, M. (2024). Advancing requirements engineering through generative ai: Assessing the role of llms. In *Generative AI for Effective Software Development*, pages 129–148. Springer.

Bajaj, D., Goel, A., Gupta, S., and Batra, H. (2022). Muce: a multilingual use case model extractor using gpt-3. *International Journal of Information Technology*, 14(3):1543–1554.

Bertram, V., Kausch, H., Kusmenko, E., Nqiri, H., Rumpe, B., and Venhoff, C. (2023). Leveraging natural language processing for a consistency checking toolchain of automotive requirements. In *2023 IEEE 31st International Requirements Engineering Conference (RE)*, pages 212–222. IEEE.

Binder, M. and Mezhuyev, V. (2024). A framework for creating an iot system specification with chatgpt. *Internet of Things*, 27:101218.

Blasek, N., Eichenmüller, K., Ernst, B., Götz, N., Nast, B., and Sandkuhl, K. (2023). Large language models in requirements engineering for digital twins. In *PoEM Companion*.

Committee, I. C. S. S. E. S. and Board, I.-S. S. (1998). *IEEE recommended practice for software requirements specifications*, volume 830. IEEE.

Davis, A., Overmyer, S., Jordan, K., Caruso, J., Dandashi, F., Dinh, A., Kincaid, G., Ledeboer, G., Reynolds, P., Sitaram, P., et al. (1993). Identifying and measuring quality in a software requirements specification. In *[1993] Proceedings First International Software Metrics Symposium*, pages 141–152. Ieee.

Dearstyne, K. R., Rodriguez, A. D., and Cleland-Huang, J. (2024). Supporting software maintenance with dynamically generated document hierarchies. In *2024 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 426–437. IEEE.

Fernández, D. M., Wagner, S., Kalinowski, M., Felderer, M., Mafra, P., Vetrò, A., Conte, T., Christiansson, M.-T., Greer, D., Lassenius, C., et al. (2017). Naming the pain in requirements engineering: Contemporary problems, causes, and effects in practice. *Empirical software engineering*, 22(5):2298–2338.

Ferrari, A., Esuli, A., Gori, M., et al. (2023). Chatgpt for requirements engineering: Threat or opportunity? In *2023 IEEE 31st International Requirements Engineering Conference (RE)*, pages 13–23. IEEE.

Ferrari, A. and Spoletini, P. (2025). Formal requirements engineering and large language models: A two-way roadmap. *Information and Software Technology*, 181:107697.

Hemmat, A., Sharbaf, M., Kolahdouz-Rahimi, S., Lano, K., and Tehrani, S. Y. (2025). Research directions for using llm in software requirement engineering: A systematic review. *Frontiers in Computer Science*, 7:1519437.

Hou, X., Zhao, Y., Liu, Y., Yang, Z., Wang, K., Li, L., Luo, X., Lo, D., Grundy, J., and Wang, H. (2024). Large language models for software engineering: A systematic literature review. *ACM Transactions on Software Engineering and Methodology*, 33(8):1–79.

Huang, K., Wang, F., Huang, Y., and Arora, C. (2025). Prompt engineering for requirements engineering: A literature review and roadmap. *arXiv preprint arXiv:2507.07682*.

ISO, I. (2018). Iec/ieee 29148: 2018 systems and software engineering-life cycle processes. *Requirements engineering*.

Jain, C., Anish, P. R., Singh, A., and Ghaisas, S. (2023). A transformer-based approach for abstractive summarization of requirements from obligations in software engineering contracts. In *2023 IEEE 31st International Requirements Engineering Conference (RE)*, pages 169–179. IEEE.

Ji, Z., Lee, N., Frieske, R., et al. (2023). Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

Kitchenham, B. and Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering.

Krishna, M., Gaur, B., Verma, A., and Jalote, P. (2024). Using llms in software requirements specifications: An empirical evaluation. In *2024 IEEE 32nd International Requirements Engineering Conference (RE)*, pages 475–483. IEEE.

Liberati, A., Altman, D. G., Tetzlaff, J., Mulrow, C., Gøtzsche, P. C., Ioannidis, J. P., Clarke, M., Devereaux, P. J., Kleijnen, J., and Moher, D. (2009). The prisma statement for reporting systematic reviews and meta-analyses of studies that evaluate healthcare interventions: explanation and elaboration. *Bmj*, 339.

Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. In *Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004)*, pages 74–81. ACL.

Liu, H., García, M. B., and Korkakakis, N. (2024). Exploring multi-label data augmentation for llm fine-tuning and inference in requirements engineering: A study with domain expert evaluation. In *2024 International Conference on Machine Learning and Applications (ICMLA)*, pages 432–439. IEEE.

Lubos, S., Felfernig, A., Tran, T. N. T., Garber, D., El Mansi, M., Erdeniz, S. P., and Le, V.-M. (2024). Leveraging llms for the quality assurance of software requirements. In *2024 IEEE 32nd International Requirements Engineering Conference (RE)*, pages 389–397. IEEE.

Nakagawa, H. and Honiden, S. (2023). Mape-k loop-based goal model generation using generative ai. In *2023 IEEE 31st International Requirements Engineering Conference Workshops (REW)*, pages 247–251. IEEE.

Nicklas, M. and Mozelius, P. (2024). Exploring student perspectives on generative ai in requirements engineering education. In *ICAIR 2024*, volume 4.

Norheim, J. J. and Rebentisch, E. (2024). Structuring natural language requirements with large language models. In *2024 IEEE 32nd International Requirements Engineering Conference Workshops (REW)*, pages 68–71. IEEE.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Pohl, K. (2010). Fundamentals, principles, and techniques.

Ramasamy, V., Ramamoorthy, S., Walia, G. S., Kulpinski, E., and Antreassian, A. (2024). Enhancing user story generation in agile software development through open ai and prompt engineering. In *2024 IEEE Frontiers in Education Conference (FIE)*, pages 1–8. IEEE.

Ronanki, K., Berger, C., and Horkoff, J. (2023). Investigating chatgpt's potential to assist in requirements elicitation processes. In *2023 49th Euromicro conference on software engineering and advanced applications (SEAA)*, pages 354–361. IEEE.

Ruan, K., Chen, X., and Jin, Z. (2023). Requirements modeling aided by chatgpt: An experience in embedded systems. In *2023 IEEE 31st International Requirements Engineering Conference Workshops (REW)*, pages 170–177. IEEE.

Sabetzadeh, M. and Arora, C. (2025). Practical guidelines for the selection and evaluation of natural language processing techniques in requirements engineering. In *Handbook on Natural Language Processing for Requirements Engineering*, pages 407–433. Springer.

Sami, M. A., Waseem, M., Zhang, Z., Rasheed, Z., Systä, K., and Abrahamsson, P. (2024). Early results of an ai multiagent system for requirements elicitation and analysis. In *International Conference on Product-Focused Software Process Improvement*, pages 307–316. Springer.

Santos, C. A. d., Bouchard, K., and Minetto Napoleão, B. (2025). Automatic user story generation: a comprehensive systematic literature review. *International Journal of Data Science and Analytics*, 20(1):1–24.

Santos, S., Breaux, T., Norton, T., Haghighi, S., and Ghanavati, S. (2024). Requirements satisfiability with in-context learning. In *2024 IEEE 32nd International Requirements Engineering Conference (RE)*, pages 168–179. IEEE.

Schwedt, S. and Ströder, T. (2025). From bugs to benefits: Improving user stories by leveraging crowd knowledge with cruise-ac. *arXiv preprint arXiv:2501.15181*.

Shah, S. T. U., Hussein, M., Barcomb, A., and Moshirpour, M. (2025). From inductive to deductive: Llms-based qualitative data analysis in requirements engineering. *arXiv preprint arXiv:2504.19384*.

Sharma, A., Chaturvedi, A., and Tripathi, A. K. (2024). From problem descriptions to user stories: Utilizing large language models through prompt chaining. In *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pages 1–6. IEEE.

Siddeshwar, V., Alwidian, S., and Makrehchi, M. (2024). A systematic review of ai-enabled frameworks in requirements elicitation. *IEEE Access*.

Sommerville, I. (2011). *Software Engineering*. Pearson Education, 9th edition.

Touvron, H., Lavril, T., Izacard, G., et al. (2023). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Umar, M. A. and Lano, K. (2024). Advances in automated support for requirements engineering: a systematic literature review. *Requirements Engineering*, 29(2):177–207.

Verdecchia, R., Engström, E., Lago, P., Runeson, P., and Song, Q. (2023). Threats to validity in software engineering research: A critical reflection. *Information and Software Technology*, 164:107329.

Vogelsang, A. (2024). From specifications to prompts: On the future of generative large language models in requirements engineering. *IEEE Software*, 41(5):9–13.

Voria, G., Casillo, F., Gravino, C., Catolino, G., and Palomba, F. (2025). Recover: Toward requirements generation from stakeholders' conversations. *IEEE Transactions on Software Engineering*.

Wagner, S., Fernández, D. M., Felderer, M., Vetrò, A., Kalinowski, M., Wieringa, R., Pfahl, D., Conte, T., Christiansson, M.-T., Greer, D., et al. (2019). Status quo in requirements engineering: A theory and a global family of surveys. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 28(2):1–48.

Wang, B., Wang, C., Liang, P., Li, B., and Zeng, C. (2024). How llms aid in uml modeling: An exploratory study with novice analysts. In *2024 IEEE International Conference on Software Services Engineering (SSE)*, pages 249–257. IEEE.

Wiegers, K. E. and Beatty, J. (2013). *Software Requirements*. Microsoft Press, 3rd edition.

Xu, Y., Feng, J., and Miao, W. (2024). Learning from failures: Translation of natural language requirements into linear temporal logic with large language models. In *2024 IEEE 24th International Conference on Software Quality, Reliability and Security (QRS)*, pages 204–215. IEEE.

Zhang, B., Carriero, V. A., Schreiberhuber, K., Tsaneva, S., González, L. S., Kim, J., and de Berardinis, J. (2024). Ontochat: a framework for conversational ontology engineering using language models. In *European Semantic Web Conference*, pages 102–121. Springer.

Zhang, T., Kishore, V., Wu, F., et al. (2020). Bertscore: Evaluating text generation with bert. *International Conference on Learning Representations (ICLR)*.

Zhao, L., Alhoshan, W., Ferrari, A., Letsholo, K. J., Ajagbe, M. A., Chioasca, E.-V., and Batista-Navarro, R. T. (2021). Natural language processing for requirements engineering: A systematic mapping study. *ACM Computing Surveys (CSUR)*, 54(3):1–41.

Zhao, W. X., Zhou, K., Li, J., et al. (2023a). A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Zhao, Z., Zhang, L., Lian, X., Gao, X., Lv, H., and Shi, L. (2023b). Reqgen: Keywords-driven software requirements generation. *Mathematics*, 11(2):332.

Zitouni, M. N., Anda, A. A., Rajpal, S., Amyot, D., and Mylopoulos, J. (2025). Towards the llm-based generation of formal specifications from natural-language contracts: Early experiments with symboleo. In *2025 IEEE/ACM Requirements Engineering for AI-powered SoftwarE (RAISE)*, pages 1–9. IEEE.