Tertiary Study about Requirements Engineering and Software for Machine Learning and Artificial Intelligence

PDC Team Squad 1[1]**, Renato de Freitas Bulcão Neto**[1]

[1]Instituto de Informática – Universidade Federal de Goiás (UFG)
Caixa Postal 131 – 74690-900 – Goiânia – GO – Brazil

Abstract. *The rapid advancement of Generative Artificial Intelligence (GenAI) has triggered growing interest in its application across Software Engineering (SE) activities. This report presents a tertiary study that systematically reviews the existing secondary studies on the use of GenAI in SE. A comprehensive search was conducted in the Scopus database, resulting in a corpus of approximately 60 relevant articles. The selected studies were analyzed to synthesize and classify the main GenAI methods, tools, and models applied throughout the software development lifecycle, as well as the reported challenges, research gaps, and directions for future work. This tertiary study provides a structured overview of the current state of research on GenAI in SE, identifies key trends and limitations, and outlines promising avenues for future investigation. The findings aim to support researchers and practitioners in understanding the maturity of the field and in guiding the responsible and effective use of GenAI in SE practice.*

1. Protocol

1.1. General Objectives

Map studies that report the use of generative AI to support software engineering activities.

1.2. Research Question

1.2.1. RQ1 - What is the state of the art of genAI for software engineering and what are the most researched topics in this area?

Identify research and proposals regarding the use of generative AI for software engineering activities. Identify the main methods, techniques, proposals, uses, evaluation methods and metrics, when and by whom they occurred, as well as the supporting tools and roles involved in the use of requirements engineering for machine learning and artificial intelligence.

1.2.2. RQ2 - What are the challenges and gaps highlighted by the genAI for software engineering activities?

Verify the risks, limitations, challenges, and possible paths for future research on gen AI for software engineering.

The figure 1 presents the study selection flow adopted in the tertiary study, similar to a SEGRESS diagram, detailing the steps of identification, screening, eligibility, and inclusion of articles.

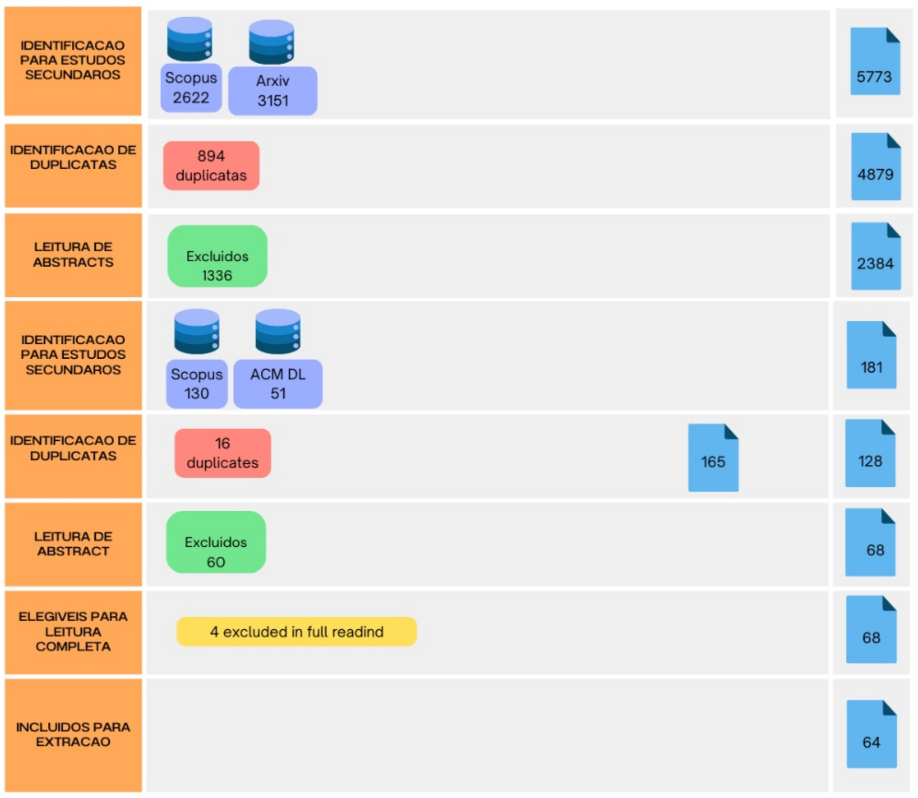
IDENTIFICACAO PARA ESTUDOS SECUNDAROS
Scopus 2622
Arxiv 3151
5773
IDENTIFICACAO DE DUPLICATAS
894 duplicatas
4879
LEITURA DE ABSTRACTS
Excluidos 1336
2384
IDENTIFICACAO PARA ESTUDOS SECUNDAROS
Scopus 130
ACM DL 51
181
IDENTIFICACAO DE DUPLICATAS
16 duplicates
165
128
LEITURA DE ABSTRACT
Excluidos 60
68
ELEGIVEIS PARA LEITURA COMPLETA
4 excluded in full readind
68
INCLUIDOS PARA EXTRACAO
64

Figura 1. Literature review process developed in this project.

Initially, in the secondary study identification phase, 5,773 records were retrieved, 2,622 from Scopus and 3,151 from arXiv. Next, duplicates were removed, resulting in the exclusion of 894 records, which reduced the set to 4,879 unique studies.

In the abstract reading stage, 1,336 studies were excluded based on the inclusion and exclusion criteria, leaving 2,384 articles. From this set, a new identification focused on secondary studies was conducted, now using the Scopus (130) and ACM Digital Library (51) databases, totaling 181 records. After a further check for duplicates, 16 duplicate studies were removed, leaving 165 records, of which 128 proceeded to the next phase. In the reading of abstracts in this stage, 60 studies were excluded, resulting in 68 articles eligible for full reading.

During the full reading, 4 studies were excluded, and, at the end of the process, 64 studies were included for data extraction.

In summary, the figure shows a rigorous and incremental filtering process, ensuring traceability and transparency in the selection of studies that support the analysis of the use of generative AI in Software Engineering.

1.3. PICOC Criteria

- **Population:** Secondary studies published about GenAI applied to Software Engineering.
- **Intervention:** Methods, tools, techniques, process, approach.
- **Comparison:** Not applicable.

- **Outcomes:** State of the art about GenAI applied to Software Engineering.
- **Context:** Without application area.

1.4. Identification of Study

1.4.1. Search Strategy

Automatin Search

1.4.2. List of Search Sourcs

- Scopus: https://www.scopus.com/
- arXiv: https://arxiv.org

1.4.3. String

(”generative AI”OR ”language model”OR LLM OR ”AI-assisted”OR ”AI-assisted software”) AND (”software engineering”OR ”requirements engineering”OR ”software architecture”OR ”software design”OR ”software testing”OR ”devops”OR ”software maintenance”OR ”code generation”OR ”test generation”) AND (”systematic review”OR ”literature survey”OR ”systematic literature review”OR ”systematic mapping”OR ”systematic literature mapping”)

1.5. Selection of Studies

Inclusion Criteria

- IC: The secondary study identifies proposals, use, or evaluation of GenAI for Software Engineering activities.

  Exclusion Criteria

- EC1: Full text not available for free on the Web or CAPES Portal
- EC2: Not written in English
- EC3: Not a secondary study
- EC4: Does not address GenAI for SE activities
- EC5: Preliminary or summarized version of another included study

1.6. Quality Evaluation

1.7. Criteria

- QC1 – Are the inclusion and exclusion criteria properly described?
- QC2 – Does the search cover all relevant studies?
- QC3 – Is the quality of included primary studies assessed?
- QC4 – Are primary studies adequately described?
- QC5 – Is the justification for the study adequately described?
- QC6 – Is the protocol validation properly described?
- QC7 – Is the extraction properly described and appropriate?

1.8. Extraction Form

- Title
- Authors
- Affiliation
- Publication vehicle
- Year of publication
- Publication type
- Search source
- Number of citations
- How many search sources
- What search strategies
- How many primary studies were analyzed
- Objective of the secondary study
- Search string used in each secondary study
- Contributions of each secondary study
- What is the state of the art of GenAI for SE and what are the most researched topics in requirements engineering for machine learning?
- What are the contributions of AI in SE studies (techniques, processes, tools, metrics)?
- What SE steps are explored in GenAI studies?
- What GenAI techniques are explored in AI or ML studies?
- What LLMs, SLMs, prompts and architectures are explored in GenAI for SE studies?
- What validation techniques are explored in GenAI for SE studies? (data, prompts, metrics and results)
- What are the tools that support GenAI for SE studies?
- What are the challenges and gaps highlighted by the requirements engineering literature for machine learning and artificial intelligence?
  What are the challenges identified in each secondary study?
  Future work highlighted in each secondary study?

2. Results

We began the literature review process with a secondary study. We constructed the protocol, conducted a pilot test, and performed a sensitivity analysis of the terms. We obtained 2,622 studies from Scopus and 3,151 studies from Arxiv, totaling 5,773 studies. After selecting studies, we obtained 2,384 studies to be read. We recognized the importance of analyzing these studies. However, given the deadline for completing our study, analysis became unfeasible.

Therefore, we proceeded to plan a tertiary study. In this case, we obtained 51 studies from Arxiv and 130 studies from Scopus, totaling 181 studies. We decided to continue with the selection and extraction of studies from Scopus. Thus, we obtained 68 studies from Scopus, which were reported in this study. Table 1 presents all studies with the study ID, article title, and reference.

Table 2 presents the study IDs, year, total study score, and the response attributed to the quality assessment question. The possible responses are: 1.0 fully attend to, 0.5

Tabela 1. List of Secondary Studies selected in this study.

ID	Title of Paper	Reference
A1	A Catalog of Data Smells for Coding Tasks	[Vitale et al. 2025]
A2	A Pilot Study on AI-Assisted Code Generation with Large Language Models	[Liu et al. 2024]
A3	A Review of Reasoning in Artificial Agents using Large Language Models	[Naidu and El-Gayar 2025]
A4	A Survey of Natural Language-Based Editing of Low-Code Applications Using Large Language Models	[Gorissen et al. 2024]
A5	A Systematic Literature Review of 10 years of Research on Program Synthesis and Natural Language Processing	[Ramírez-Rueda et al. 2024]
A6	A Systematic Literature Review of Large Language Model Applications in Industry	[Moenks et al. 2025]
A7	A Systematic Literature Review on Using Natural Language Processing in Software Requirements Engineering	[Necula et al. 2024]
A8	A Systematic Mapping Study of LLM Applications in Mobile Device Research	[Chen et al. 2025a]
A9	A Systematic Review of AI-Enabled Frameworks in Requirements Elicitation	[Siddeshwar et al. 2024]
A10	A systematic literature review on logging smell detection	[Madi and Binkhonain 2026]
A11	AI and Teamwork in Agile Software Development: A Systematic Mapping Study	[Kwok and Adil 2025]
A12	AI-Based Approaches for Software Tasks Effort Estimation: A Systematic Review of Methods and Trends	[Rossi and Fontoura 2025]
A13	AI-assisted Software Engineering: A tertiary study	[Cico et al. 2023]
A14	AI-enabled Software Engineer: A Taxonomy of Challenges and Success Factors	[Shameem et al. 2024]
A15	Agent design pattern catalogue: A collection of architectural patterns for foundation model based agents	[Liu et al. 2025]
A16	Artificial intelligence in web accessibility: A systematic mapping study	[Abou-Zahra et al. 2018]
A17	Assessing the Effectiveness of ChatGPT in Secure Code Development: A Systematic Literature Review	[Bouzid and Khoury 2025]
A18	Automatic Extraction and Formalization of Temporal Requirements from Text: A Survey	[Barrientos et al. 2025]
A19	Automating Code-Related Tasks Through Transformers: The Impact of Pre-training	[Tufano et al. 2023]
A20	Beyond the Systematic: Forecasting Importance and Emergence of Research Areas in Applications of Software Traceability Using NLP	[Pauzi and Capiluppi 2024]
A21	Code Generation Using Machine Learning: A Systematic Review	[Dehaerne et al. 2022]
A22	Decoding ChatGPT: A taxonomy of existing research, current challenges, and possible future directions	[Sohail et al. 2023]
A23	Effectiveness of Generative Artificial Intelligence for Scientific Content Analysis	[Platt and Platt 2023]
A24	Empowering business transformation: The positive impact and ethical considerations of generative AI in software product management	[Parikh 2023]
A25	Enhancing Software Requirements Engineering with Language Models and Prompting Techniques: Insights from the Current Research and Future Directions	[Ebrahim et al. 2025]
A26	Exploring the role of generative AI in enhancing cybersecurity in software development life cycle	[Al-Hashimi et al. 2025]
A27	Generating and Reviewing Programming Codes with Large Language Models: A Systematic Mapping Study	[Albuquerque et al. 2024]
A28	Generative AI for Requirements Engineering: A Systematic Literature Review	[Cheng et al. 2025]
A29	Generative Artificial Intelligence and Large Language Models: A Systematic Review of Architectures, Applications, and Future Directions	[Ciubotaru 2025]
A30	Generative Artificial Intelligence in Agile Software Development Processes: A Literature Review Focused on User eXperience	[Cornide-Reyes et al. 2025]
A31	How Well Small Language Models Can Be Adapted for Software Maintenance and Refactoring Tasks	[Asvydyte et al. 2025]
A32	How reliance on GenAI might limit human creativity and critical thinking in Requirements Engineering	[A32 2025]
A33	Impact of Artificial Intelligence on Open-Source Software Development	[Ahuja et al. 2025]
A34	Impacts of the Usage of Generative Artificial Intelligence on Software Development Process	[Santos et al. 2024]
A35	Insight into code clone management through refactoring: a systematic literature review	[Kaur et al. 2025]
A36	LLM-Based Code Generation: A Systematic Literature Review With Technical and Demographic Insights	[Umama et al. 2025]
A37	LLM-Based Multi-Agent Systems for Software Engineering: Literature Review, Vision, and the Road Ahead	[He et al. 2025]
A38	Landscape and Taxonomy of Prompt Engineering Patterns in Software Engineering	[Sasaki et al. 2025a]
A39	Large Language Model for Vulnerability Detection and Repair: Literature Review and the Road Ahead	[Sasaki et al. 2025b]
A40	Large Language Models for Early-Stage Software Project Estimation: A Systematic Mapping Study	[Radliński and Swacha 2025]
A41	Large Language Models for Software Engineering: A Systematic Literature Review	[Hou et al. 2024]
A42	Large Language Models for Service-Oriented Computing (LLM4SOC): Review and Research Directions	[Kumara et al. 2025]
A43	Large Language Models for Software Engineering: A Systematic Mapping Study	[Görmez et al. 2024]
A44	Learning-Based Automated Program Repair: A Systematic Literature Review	[Chen et al. 2025b]
A45	Leveraging AI-driven requirements for SysML modeling of the IoBT: A comprehensive investigation	[A45 2025]
A46	Machine Learning Techniques for Automatic Program Repair: A Systematic Literature Mapping	[Domínguez-Isidro et al. 2025]
A47	Machine learning approaches for automated software traceability: A systematic literature review	[Alturayeif et al. 2025]
A48	Machine learning for requirements engineering (ML4RE): A systematic literature review complemented by practitioners' voices from Stack Overflow	[Li et al. 2024]
A49	Prompt Engineering Guidelines for Using Large Language Models in Requirements Engineering	[Ronanki et al. 2025]
A50	Prompt Engineering for Requirements Engineering: A Literature Review and Roadmap	[Huang et al. 2025]
A51	Prompting Techniques for Secure Code Generation: A Systematic Investigation	[Tony et al. 2025]
A52	Prompts Engineering Challenges in Software Code Generation	[Gutierrez et al. 2025]
A53	Psycholinguistic analyses in software engineering text: A systematic mapping study	[Sajadi et al. 2026]
A54	Quality Assurance for LLM-Generated Test Cases: A Systematic Literature Review	[Edirisinghe and Wickramaarachchi 2024]
A55	Research directions for using LLM in software requirement engineering: a systematic review	[Hemmat et al. 2025]
A56	Reusable and generic design decisions for developing UML-based domain-specific languages	[Hoisl et al. 2017]
A57	Software Engineering and Foundation Models: Insights from Industry Blogs Using a Jury of Foundation Models	[Li et al. 2025]
A58	State of the Art of the Security of Code Generated by LLMs: A Systematic Literature Review	[Ramírez et al. 2024]
A59	Systematic Literature Review of Prompt Engineering Patterns in Software Engineering	[Sasaki et al. 2024]
A60	Talking to Data: A Systematic Review of the Rise of Conversational Agents for Visual Analytics	[Martins et al. 2025]
A61	The Role of Generative AI Models in Requirements Engineering: A Systematic Literature Review	[Vasudevan and Reddivari 2025]
A62	The use of large language models for program repair	[Zubair et al. 2025]
A63	Time-Series Large Language Models: A Systematic Review of State-of-the-Art	[Abdullahi et al. 2025]
A64	Towards Using Personas in Requirements Engineering: What Has Been Changed Recently?	[Muzammel et al. 2025]
A65	Unlocking the Potential of the Prompt Engineering Paradigm in Software Engineering: A Systematic Literature Review	[Syahputri et al. 2025]
A66	Using LLMs to enhance code quality: A systematic literature review	[Alomari et al. 2026]
A67	Using Reinforcement Learning for Security Testing: A Systematic Mapping Study	[Ahmad et al. 2025]
A68	Why Adapt RAG for Agile? Challenges, Frameworks, and the Role of Evaluator Agent	[Khan et al. 2025]

partially attend to, and 0.0 not attend to. These were possible responses for each of the 7 questions. Items without a quality assessment question response were studies that were excluded because they did not fall within the expected scope of the analysis, as foreseen in the exclusion criteria.

2.1. Used Models Description:

Models that were in the checkbox list at the extraction form: The family GPT-X was covered in 29 articles: A1, A5, A6, A7, A10, A12, A17, A18, A22, A24, A25, A27, A30, A36, A38, A40, A44, A45, A48, A50, A52, A54, A55, A58, A59, A60, A61, A62 and A67. The Llama model was cited in 12 articles: A1, A6, A25, A27, A36, A40, A50, A55, A60, A64, A65 and A66. The TinyLama appeared in 2 articles: A25 and A66. The Claude model was mentioned in 4 articles: A6, A36, A40, and A50. DeepSeek was cited in 2 articles: A6 and A50. The Mistral model appeared in 4 articles: A40, A50, A60, and A66. The StarCoder model was discussed in 6 articles: A17, A27, A36, A61, A62, and A66. The Phi model was cited in 1 article: A50. The “Other” option (a multiple-choice option in the extraction form) was recorded in 22 articles: A1, A5, A7, A9, A10, A17, A18, A19, A21, A22, A24, A27, A30, A36, A48, A54, A55, A57, A58, A60, A62, and A66.

In addition, there were 28 citations of additional models in the articles (not present as an option in the multiple-choice question) A1, A10, A17, A18, A21, A22, A24, A25, A27, A30, A36, A37, A38, A40, A41, A44, A48, A50, A54, A55, A57, A58, A59, A60, A61, A62, A66, and A68, as detailed below. The Log Sculptor and LogUpdater models were cited exclusively in article A10.

The CodeGen model appeared in 7 articles: A17, A27, A36, A38, A41, A59, and A62. InCoder was cited in 3 articles: A17, A27, and A62. The BERT family of models was discussed in 13 articles: A17, A18, A22, A27, A40, A41, A44, A48, A50, A59, A61, A62, and A66. BART appeared in 1 article: A41. GitHub Copilot was cited in 2 articles: A17 and A30. Amazon Code Whisperer was cited in 1 article: A17. The Bard model was mentioned in 3 articles: A17, A24, and A50. The StackRAG model was cited in 3 articles: A41, A50, and A68. The Codex model appeared in 13 articles: A22, A24, A27, A36, A38, A41, A44, A54, A55, A58, A59, A62, and A66. The PaLM model was cited in 2 articles: A41 and A66. The ELMo appeared in 1 article: A66. The UniXcoder model was cited in 2 articles: A27 and A62. The T5 model was discussed in 7 articles: A36, A38, A41, A44, A50, A59, and A61. The Pegasus was cited in 1 article: A61. MiniLM appeared in 1 article: A61. The Vicuna model was cited in 1 article: A60. Gemini was mentioned in 4 articles: A37, A40, A50 and A60. Pro/Vision was cited in 1 article: A60. The PolyCoder model was cited in 2 articles: A38 and A59. The XLNet model was cited in 1 article: A50. Nous-Hermes-2 was cited in 1 article: A50. The CodeGeeX model was cited in 1 article: A50. ERNIE Bot was cited in 1 article: A50. The Whisper model was cited in 1 article: A50. The davinci model was mentioned in 2 articles: A50 and A66. The Word2Vec model was cited in 1 article: A48. The AlphaCode model was cited in 1 article: A27. Tabnine was mentioned in 1 article: A30. The Cypress Copilot was mentioned in 2 articles: A27 and A30. The LSTM model appeared in 3 articles: A44, A48, and A66. Finally, the MiniCPM model was mentioned in 1 article: A25.

Tabela 2. Quality evaluation of secundary studies.

ID	Ano	Score Total	Q1	Q2	Q3	Q4	Q5	Q6	Q7
A1	2025	5.0	Partially attended to	Fully attended to	Not attended to	Fully attended to	Fully attended to	Partially attended to	Fully attended to
A2	2024	0.0	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to
A3	2025	0.0	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to
A4	2024	0.0	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to
A5	2024	6.5	Partially attended to	Fully attended to	Fully attended to	Fully attended to	Fully attended to	Fully attended to	Fully attended to
A6	2025	6.0	Fully attended to	Fully attended to	Partially attended to	Partially attended to	Fully attended to	Fully attended to	Fully attended to
A7	2024	6.0	Fully attended to	Partially attended to	Fully attended to	Fully attended to	Fully attended to	Fully attended to	Partially attended to
A8	2025	0.0	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to
A9	2024	6.0	Fully attended to	Fully attended to	Partially attended to	Fully attended to	Fully attended to	Partially attended to	Fully attended to
A10	2026	4.5	Not attended to	Fully attended to	Not attended to	Partially attended to	Fully attended to	Fully attended to	Fully attended to
A11	2026	5.0	Partially attended to	Fully attended to	Not attended to	Partially attended to	Fully attended to	Fully attended to	Fully attended to
A12	2025	5.0	Partially attended to	Fully attended to	Partially attended to	Partially attended to	Fully attended to	Partially attended to	Fully attended to
A13	2023	0.0	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to
A14	2024	0.0	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to
A15	2025	0.0	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to
A16	2026	5.5	Partially attended to	Fully attended to	Fully attended to	Partially attended to	Fully attended to	Partially attended to	Fully attended to
A17	2025	5.5	Partially attended to	Partially attended to	Fully attended to	Fully attended to	Fully attended to	Partially attended to	Fully attended to
A18	2025	6.0	Fully attended to	Fully attended to	Partially attended to	Fully attended to	Fully attended to	Partially attended to	Fully attended to
A19	2023	3.0	Not attended to	Not attended to	Not attended to	Not attended to	Fully attended to	Fully attended to	Fully attended to
A20	2024	0.0	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to
A21	2022	6.0	Partially attended to	Fully attended to	Fully attended to	Fully attended to	Fully attended to	Partially attended to	Fully attended to
A22	2023	0.0							
A23	2023	0.5	Not attended to	Not attended to	Not attended to	Not attended to	Partially attended to	Not attended to	Not attended to
A24	2023	5.0	Partially attended to	Fully attended to	Not attended to	Partially attended to	Fully attended to	Fully attended to	Fully attended to
A25	2025	4.0	Fully attended to	Partially attended to	Not attended to	Fully attended to	Fully attended to	Not attended to	Partially attended to
A26	2025	0.0							
A27	2024	6.0	Fully attended to	Fully attended to	Fully attended to	Partially attended to	Fully attended to	Fully attended to	Partially attended to
A28	2025	0.0							
A29	2025	0.0	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to
A30	2025	5.5	Fully attended to	Fully attended to	Partially attended to	Partially attended to	Fully attended to	Fully attended to	Partially attended to
A31	2026	0.0							
A32	2025	0.0							
A33	2025	3.0	Fully attended to	Not attended to	Not attended to	Partially attended to	Partially attended to	Partially attended to	Partially attended to
A34	2024	5.5	Fully attended to	Fully attended to	Not attended to	Partially attended to	Fully attended to	Fully attended to	Fully attended to
A35	2025	0.0	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to
A36	2025	6.0	Fully attended to	Fully attended to	Not attended to	Fully attended to	Fully attended to	Fully attended to	Fully attended to
A37	2025	4.0	Partially attended to	Fully attended to	Not attended to	Fully attended to	Fully attended to	Not attended to	Partially attended to
A38	2025	3.0	Partially attended to	Partially attended to	Not attended to	Partially attended to	Fully attended to	Not attended to	Partially attended to
A39	2025	6.5	Fully attended to	Fully attended to	Fully attended to	Partially attended to	Fully attended to	Fully attended to	Fully attended to
A40	2025	7.0	Fully attended to	Fully attended to	Fully attended to	Fully attended to	Fully attended to	Fully attended to	Fully attended to
A41	2024	7.0	Fully attended to	Fully attended to	Fully attended to	Fully attended to	Fully attended to	Fully attended to	Fully attended to
A42	2026	0.0							
A43	2024	0.0	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to
A44	2025	5.0	Partially attended to	Fully attended to	Partially attended to	Partially attended to	Fully attended to	Partially attended to	Fully attended to
A45	2025	4.5	Partially attended to	Fully attended to	Partially attended to	Partially attended to	Fully attended to	Partially attended to	Partially attended to
A46	2025	0.0	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to
A47	2025	0.0	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to
A48	2024	7.0	Fully attended to	Fully attended to	Fully attended to	Fully attended to	Fully attended to	Fully attended to	Fully attended to
A49	2026	6.0	Fully attended to	Fully attended to	Fully attended to	Fully attended to	Partially attended to	Partially attended to	Fully attended to
A50	2025	6.5	Fully attended to	Fully attended to	Partially attended to	Fully attended to	Fully attended to	Fully attended to	Fully attended to
A51	2025	6.0	Fully attended to	Fully attended to	Partially attended to	Fully attended to	Partially attended to	Fully attended to	Fully attended to
A52	2025	4.5	Fully attended to	Fully attended to	Not attended to	Partially attended to	Fully attended to	Partially attended to	Partially attended to
A53	2026	0.0	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to
A54	2024	5.5	Fully attended to	Fully attended to	Not attended to	Partially attended to	Fully attended to	Fully attended to	Fully attended to
A55	2025	5.0	Fully attended to	Fully attended to	Not attended to	Partially attended to	Fully attended to	Fully attended to	Partially attended to
A56	2017	0.0	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to
A57	2025	3.0	Not attended to	Not attended to	Not attended to	Not attended to	Fully attended to	Fully attended to	Fully attended to
A58	2024	6.0	Partially attended to	Fully attended to	Partially attended to	Fully attended to	Fully attended to	Fully attended to	Fully attended to
A59	2024	5.5	Fully attended to	Fully attended to	Partially attended to	Partially attended to	Fully attended to	Fully attended to	Partially attended to
A60	2025	7.0	Fully attended to	Fully attended to	Fully attended to	Fully attended to	Fully attended to	Fully attended to	Fully attended to
A61	2025	4.5	Fully attended to	Fully attended to	Not attended to	Partially attended to	Fully attended to	Partially attended to	Partially attended to
A62	2025	5.0	Partially attended to	Fully attended to	Partially attended to	Fully attended to	Fully attended to	Partially attended to	Partially attended to
A63	2025	0.0	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to	Not attended to
A64	2025	6.0	Fully attended to	Fully attended to	Partially attended to	Fully attended to	Fully attended to	Partially attended to	Fully attended to
A65	2025	5.5	Partially attended to	Fully attended to	Not attended to	Fully attended to	Fully attended to	Fully attended to	Fully attended to
A66	2026	6.0	Fully attended to	Fully attended to	Fully attended to	Partially attended to	Fully attended to	Partially attended to	Fully attended to
A67	2025	5.0	Partially attended to	Partially attended to	Partially attended to	Fully attended to	Fully attended to	Partially attended to	Fully attended to
A68	2026	4.5	Not attended to	Fully attended to	Not attended to	Partially attended to	Fully attended to	Fully attended to	Fully attended to

2.2. Description of the techniques covered:

The Prompt Engineering technique was covered in 28 articles: A5, A6, A7, A11, A17, A18, A22, A24, A25, A27, A38, A39, A41, A44, A45, A49, A52, A54, A55, A58, A59, A60, A61, A62, A64, A65 and A66. The Chain of Thought technique was cited in 13 articles: A6, A17, A25, A27, A36, A38, A44, A50, A51, A52, A60, A65 and A66. The Multiagent approach appeared in 10 articles: A6, A30, A36, A52, A54, A57, A59, A60, A67 and A68. The Self Refine technique was mentioned in 6 articles: A17, A45, A51, A60, A66, and A68. The RAG (Retrieval-Augmented Generation) technique was cited in 5 articles: A25, A57, A60, A65, and A68. The ReAct approach was recorded in 3 articles: A38, A50, and A60. The "Other" option (a multiple-choice option in the extraction form) appeared in 22 articles: A1, A4, A6, A7, A9, A10, A12, A17, A18, A19, A24, A25, A27, A30, A36, A44, A48, A55, A58, A59, A62, and A66.

Furthermore, here are other techniques described that were not options in the multiple-choice question on the extraction form. Data preprocessing was cited in article A1, while post-processing of the generated code appeared in A58. The fine-tuning technique was addressed in ten articles: A1, A4, A18, A22, A41, A44, A55, A59, A61, and A62. Inductive and deductive synthesis was mentioned in A5. Natural Language Processing (NLP) appeared in eight articles: A5, A7, A9, A10, A11, A30, A45, and A60. Zero-shot approaches were reported in nine articles: A22, A25, A27, A38, A40, A51, A59, A61, and A66, while few-shot approaches appeared in seven: A38, A40, A51, A52, A59, A61, and A66. The one-shot and k-shot variations were mentioned in one article each, A51 and A50 respectively. Grammar-guided genetic programming was cited in A5. Automated testing appeared in four articles: A6, A62, A66, and A67, and the use of agents was also mentioned in four: A6, A11, A60, and A68. The term machine learning occurred in five articles: A9, A10, A11, A45, and A48, while deep learning was cited in four: A7, A9, A48, and A66. Clustering was mentioned in A7, and assistants in A11. Bayesian networks appeared in two articles: A10 and A12, and pretraining was cited in A19. Reinforcement learning was addressed in two articles: A36 and A67. The Least-to-Most Prompting and Instruction Tuning techniques were mentioned in A66. Transfer Learning appeared in four papers: A27, A44, A61, and A66. Knowledge Graphs were cited in A64, and Neural Machine Translation (NMT) in A62. Prompt-based code generation was also mentioned in A62, while Parameter Adjustment appeared in A59. Interactive Prompting was cited in two papers: A38 and A59, and Iterative Refinement in four: A44, A45, A58, and A59. LoRA was mentioned in A55 and A57, AI Chaining in A52, and PEFT in A36 and A52. Progressive hint appeared in A51. Random Forest was cited in A48, LSTM in A44, and Knowledge Distillation in A36. In-Context Learning was mentioned in A59 and A66, Tokenization in A27, and finally, Neural networks were cited in seven papers: A1, A7, A9, A21, A44, A48, and A66.

2.3. Access type:

Regarding the type of access used in the model, two main occurrences were analyzed in the studies: commercial access was cited by 16 articles: (A5, A6, A7, A16, A17, A18, A27, A30, A39, A40, A41, A44, A51, A54, A59, A60), while open source access was cited by 13 articles: (A6, A7, A9, A16, A17, A18, A27, A40, A44, A59, A61, A67, A60). In addition, there were also 7 studies that used another type of access: (A12, A19, A49, A48, A61, A67, A68).

3. RQ Responses

3.1. RQ1 - What is the state of the art of genAI for software engineering and what are the most researched topics in this area?

About the main contributions of genAI applied to SE according the areas described in SWEBOK [Committee et al. 2022].

3.1.1. Methods

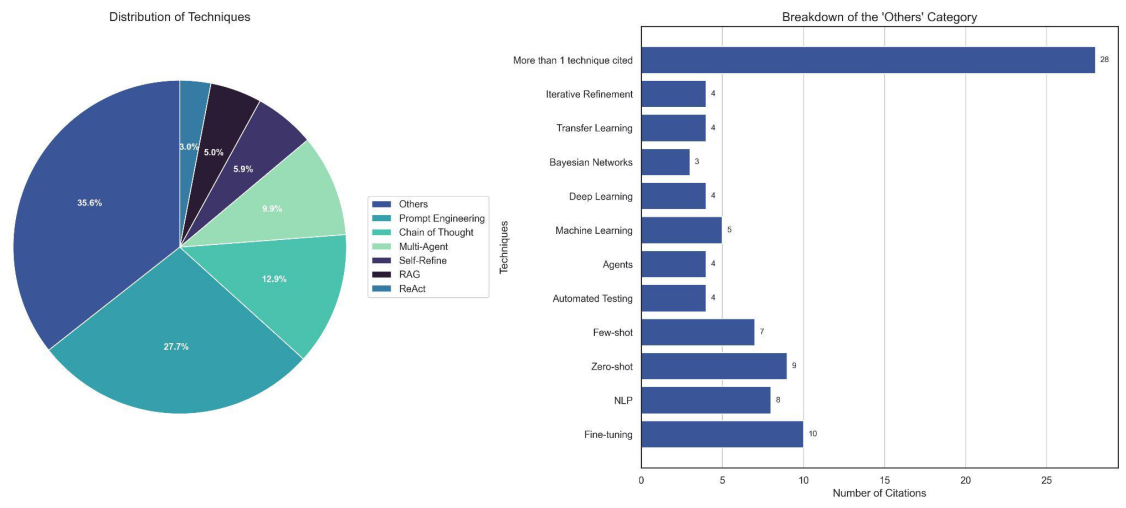

Figura 2. Distribution of Techniques.

The state-of-the-art of the methods reveals three main findings: 1 - the centrality of prompt engineering as the primary control mechanism; 2 - a transition to composite and iterative architectures; 3 - and a relative reduction in the exclusive reliance on heavy training techniques in favor of inference-time adaptation. The analysis of the 68 articles reveals a progressive consolidation of prompt engineering techniques as the central axis of language model-based approaches.

Prompt Engineering was the most frequently used technique (28 articles - A5, A6, A7, A11, A17, A18, A22, A24, A25, A27, A38, A39, A41, A44, A45, A49, A52, A54, A55, A58, A59, A60, A61, A62, A64, A65 and A66), indicating that input control and structuring remain the predominant mechanism for guiding model behavior. This data suggests that, despite advances in adaptation and fine-tuning techniques, strategic prompt manipulation remains the primary operational control interface.

The Chain of Thought (CoT) technique, present in 13 studies (A6, A17, A25, A27, A36, A38, A44, A50, A51, A52, A60, A65 and A66), reinforces the importance of strategies that promote explicit reasoning and task decomposition. Its lower frequency compared to Prompt Engineering, however, is still significant, indicating a transition from the simple formulation of instructions to approaches that structure the model's reasoning.

The emergence of the Multiagent paradigm (10 articles - A6, A30, A36, A52, A54, A57, A59, A60, A67 and A68) demonstrates a relevant architectural trend: the replacement of monolithic interactions with systems composed of specialized agents. This movement is consistent with the need for modularization, scalability, and greater robustness in complex applications.

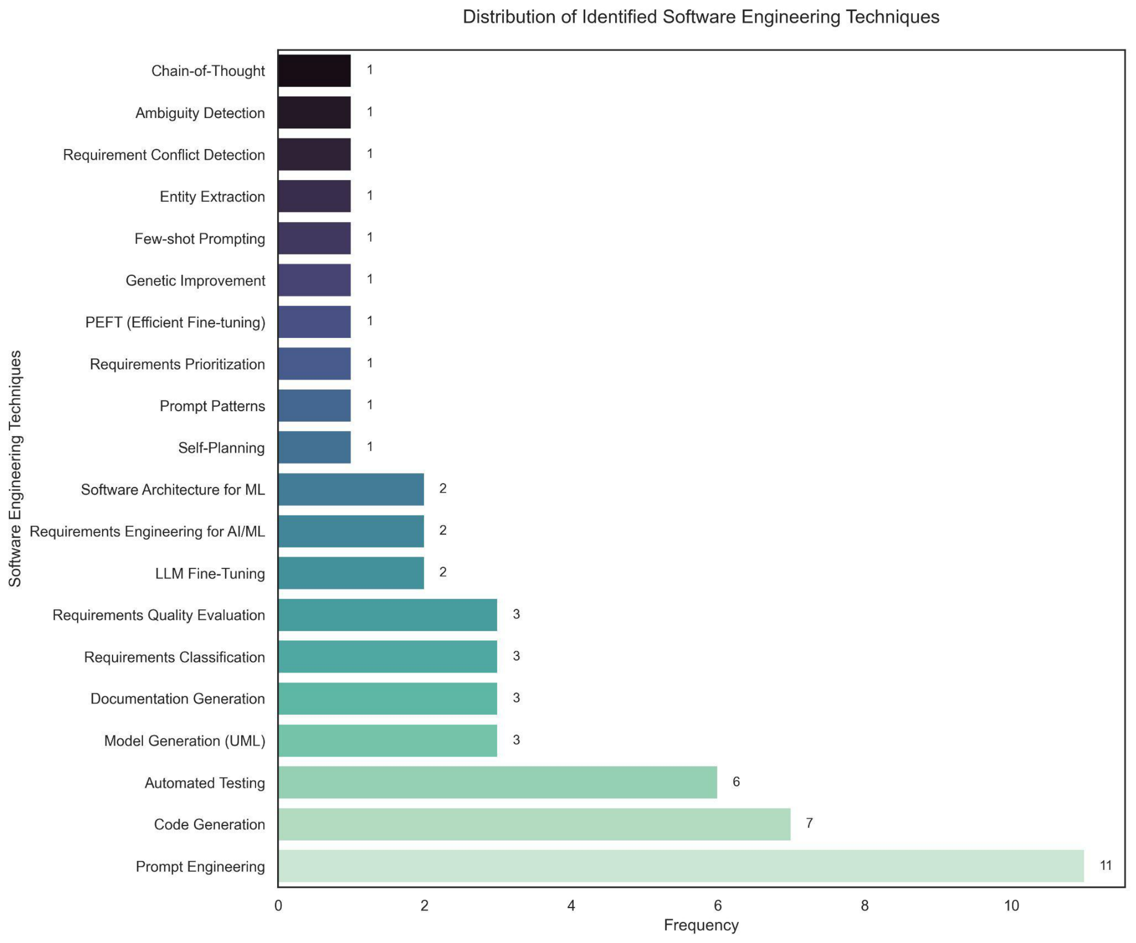

Figura 3. Distribution of identified Software Engeneering Techniques.

Iterative techniques such as Self-Refine (6 articles - A17, A45, A51, A60, A66, and A68), Iterative Refinement (4 articles - A44, A45, A58, and A59), and Interactive Prompting (2 articles - A38 and A59) demonstrate an important conceptual shift: the recognition that single-pass generation is insufficient for more complex tasks. A movement towards feedback loops and incremental refinement is observed.

The relatively lower adoption of RAG (5 articles - A25, A57, A60, A65, and A68) and ReAct (3 articles - A38, A50, and A60) suggests that hybrid approaches combining information retrieval or reasoning with action are still in the consolidation phase, despite their strategic potential to reduce hallucinations and increase factual grounding.

Furthermore, classic machine learning and deep learning techniques—such as fine-tuning (10 articles - A1, A4, A18, A22, A41, A44, A55, A59, A61, and A62), transfer learning (4 articles - A27, A44, A61, and A66), reinforcement learning (2 articles - A36 and A67), neural networks (7 articles - A1, A7, A9, A21, A44, A48, and A66), and LSTM (1 article - A44)—remain present, but less frequently than prompting-based strategies. This suggests a paradigm shift: from structural model modification to behavioral control via interaction.

The presence of zero-shot approaches (9 articles - A22, A25, A27, A38, A40, A51, A59, A61 and A66), few-shot approaches (7 articles - A38, A40, A51, A52, A59, A61 and A66), and one-shot and k-shot variations reinforces the importance of in-context learning as an alternative to extensive retraining, which is expensive and difficult. This pattern confirms the centrality of In-Context Learning, explicitly cited in two articles, as the dominant operational mechanism in contemporary LLMs.

These trends suggest that current development is less focused on modifying base models and more concentrated on the intelligent orchestration of interactions, context, and task decomposition. The field is therefore moving towards engineering distributed cognitive systems, rather than just parametric optimization of isolated models. As a future implication, greater conceptual standardization, consolidation of hybrid architectures (RAG + agents + refinement), and methodological formalization for comparative evaluation of these techniques are expected.

The evaluation techniques used in secondary studies demonstrate that the analyzed literature shows a strong tendency towards the adoption of hybrid evaluation approaches, combining quantitative metrics, automated tests, and human evaluation, with emphasis on the co-occurrence between objective metrics and human judgment. A high incidence of automated tests and generic benchmarks is also observed, indicating a concern with technical rigor, reproducibility, and comparability of results. In addition, the recurrence of consolidated metrics, such as BLEU and CodeBLEU, highlights the influence of approaches originating from natural language processing in the evaluation of code generation, reinforcing a methodological trend already consolidated in the area.

In order to improve the analysis of secondary studies, the identified evaluation techniques were organized into four categories: quantitative metrics, human evaluation, automated tests/technical analysis, and benchmarks/standard datasets. This categorization made it possible to highlight the predominance of certain approaches widely adopted in the literature. In the context of human assessment, manual assessment stood out, being used in 8 studies (A17, A21, A36, A37, A38, A51, A65, A68), followed by expert-led assessment, identified in 5 studies (A25, A45, A55, A61, A66). Among the benchmarks and datasets, a higher incidence of generic benchmarks was observed, present in 10 studies (A1, A6, A17, A27, A41, A58, A60, A62, A65, A66), while comparison with baselines was reported in 3 studies (A41, A62, A66).

In the group of quantitative metrics, the BLEU and CodeBLEU metrics were identified in 7 studies (A19, A24, A36, A44, A50, A65, A68), followed by the precision metric, recorded in 6 occurrences (A6, A9, A24, A36, A55, A65). Finally, in the category of automated tests, unit tests and test cases stood out, being adopted in 10 secondary studies (A5, A6, A16, A27, A36, A37, A44, A51, A60, A62), while static analysis appeared in 5 studies (A17, A21, A27, A51, A58).

In addition, the analysis using a co-occurrence heatmap revealed a strong concentration in the combination of quantitative metrics and human evaluation, present in 10 studies. Next, the associations between automated testing and human evaluation, automated testing and quantitative metrics, as well as benchmarks and human evaluation, were highlighted, each with 8 occurrences in the analyzed studies.

3.1.2. Metrics

Based on the analysis of selected secondary studies, the main metrics used in the evaluation processes were identified. The predominance of metrics related to quality, precision, and functional correctness is observed, present in 27 studies (A5, A6, A7, A12,

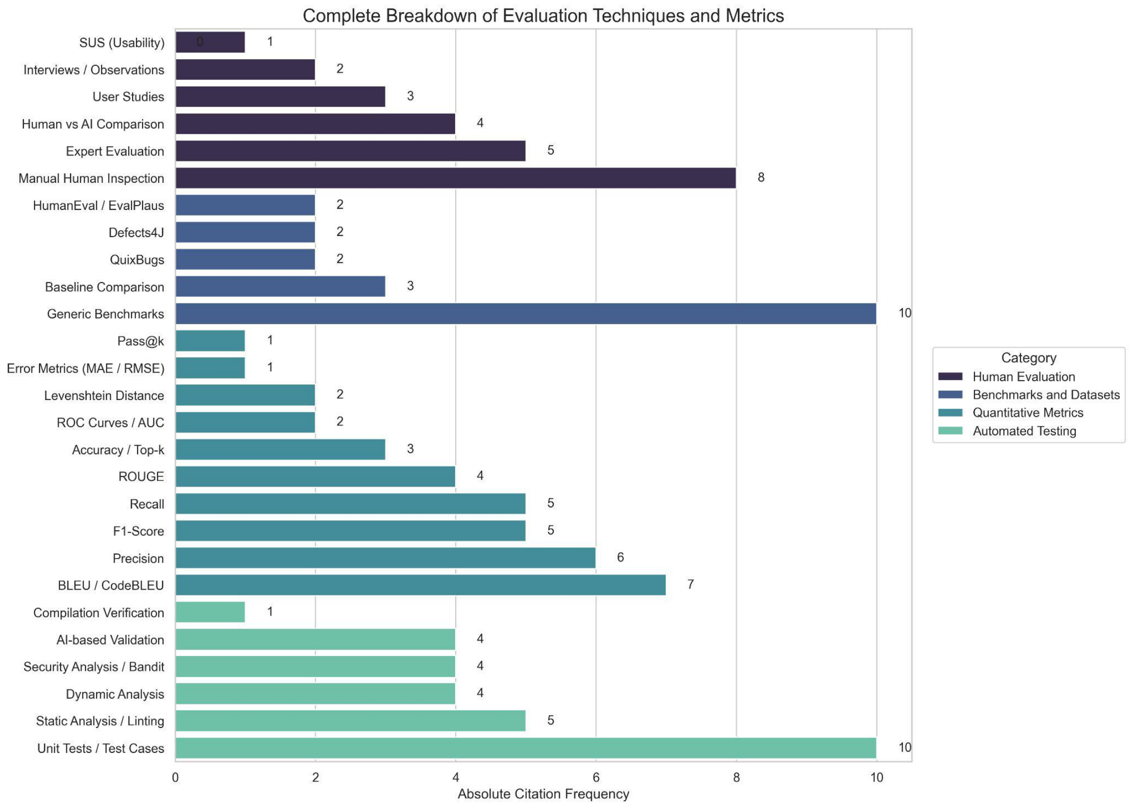

Figura 4. Complete breakdown of Evaluation Techniques and Metrics.

A16, A17, A18, A22, A24, A27, A30, A33, A36, A37, A38, A41, A48, A49, A54, A59, A60, A61, A65, A68), 24 studies (A5, A6, A7, A10, A12, A16, A17, A22, A24, A27, A30, A36, A37, A41, A44, A51, A52, A54, A58, A59, A60, A61, A62, A65, A66, A67, A68), and 16 studies (A7, A16, A17, A18, A19, A22, A27, A36, A37, A41, A51, A59, A60, A62, A67, A68), respectively. These data indicate a consolidated trend in the literature to prioritize the measurement of the technical performance of the methods evaluated.

On the other hand, metrics focused on the evaluation of human factors — such as cognitive aspects and perceived effort — are significantly less frequent, being identified in only 3 (A12, A30, A60) and 8 secondary studies (A6, A7, A22, A27, A30, A37, A67, A68), respectively. This discrepancy highlights a gap in the literature regarding the incorporation of human dimensions in the evaluation of the approaches analyzed, suggesting the need for future investigations that address these aspects more systematically.

3.1.3. Uses and Domain

The state-of-the-art analysis revealed a significant number of studies proposing the use of GenAI for the construction/coding stages (29 papers - A1, A5, A6, A11, A16, A17, A19, A21, A22, A24, A27, A30, A33, A36, A37, A38, A39, A41, A44, A51, A52, A57, A58, A59, A60, A62, A65, A66, A68) and requirements (21 papers - A5, A6, A7, A9, A11, A18, A24, A25, A27, A30, A37, A40, A41, A45, A48, A49, A50, A55, A61, A64, A68). Consequently, there is a noticeable maturity in software development applications at the code and requirements engineering levels.

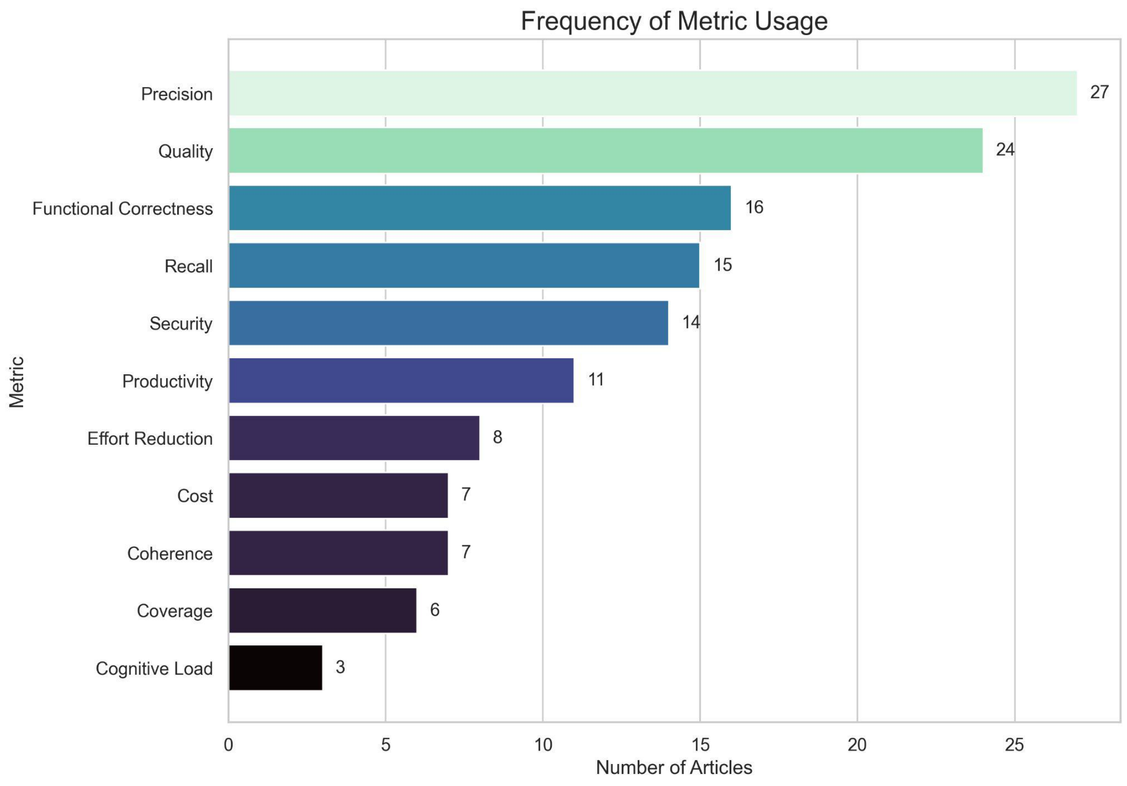

Figura 5. Frequency of metric usage.

Furthermore, a lower frequency of studies was identified involving the other stages of the SDLC, such as architecture (8 papers - A11, A16, A17, A30, A37, A41, A60, A68), operations/DevOps (7 papers - A6, A10, A11, A33, A37, A67, A68), and project management (9 papers - A5, A11, A12, A24, A33, A37, A40, A41, A68).

The discrepancies between the number of studies applying GenAI to coding and requirements versus those focusing on architecture, DevOps, and project management reveal a greater concern with understanding what is to be developed (requirements) and with code-level development (coding), rather than with more strategic, organizational, and long-term stages of the software lifecycle. In particular, activities such as architectural definition, project planning, and continuous operations (DevOps) require greater domain contextualization, integration across multiple artifacts, and decision-making under technical and organizational constraints, which may explain the lower adoption and maturity of GenAI-based approaches in these stages.

3.2. RQ2 - What are the challenges and gaps highlighted by the genAI for software engineering activities?

Regarding the main results of genAI applied to SE considering benefits, limitations, gaps, challenges and future works we can highlighted:

3.2.1. Benefits

Initially, a specific classification was performed for each benefit, consolidating 96 specific categories with their respective studies

After this, a general classification was performed, grouping the studies in a more closely overlapping way. Therefore, the studies were grouped into 9 general categories. Benefits that do not fit the 8 categories were grouped into an additional category called "Specific Benefits".

Through this synthesis, it was concluded that the most discussed benefits were "Automation of repetitive tasks" and "Increased productivity".

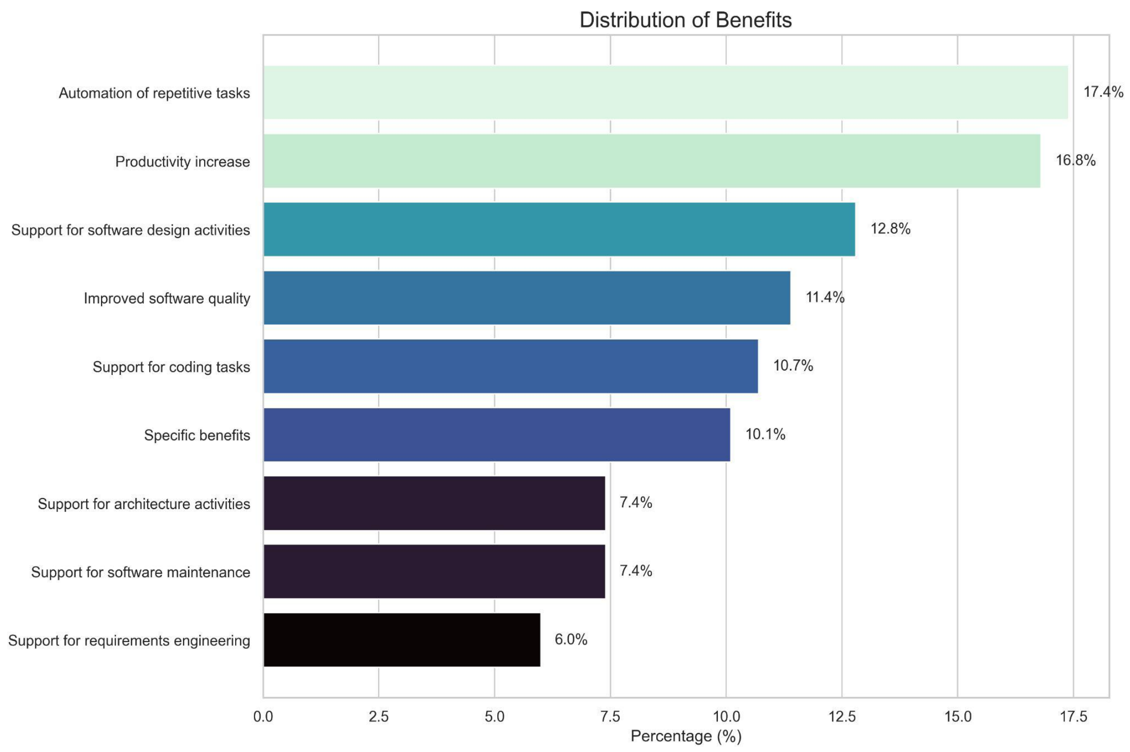

Figura 6. Distribution of benefits of GenAI in Software Engeneering.

By grouping some benefits into a "Specific Benefits" category, it was necessary to identify how these benefits were distributed. For this, a pie chart was constructed that would allow analyzing this distribution.

The conclusion reached was that the benefit "Democratization of software development" appeared more dominant in relation to the others, since 40% of the studies in this category fall into this benefit, while the remainder have the same 6.7% of studies.

In order to understand how the benefits are distributed for each stage of the software development cycle (SDLC), another grouping was made, in which the categories of automation and quality increase were taken, passing the sub-benefits to the respective stages of the SDLC.

The conclusion reached was that the benefits are mainly found in the Software Design and Coding stages, with, respectively, 25% and 23.8% of studies

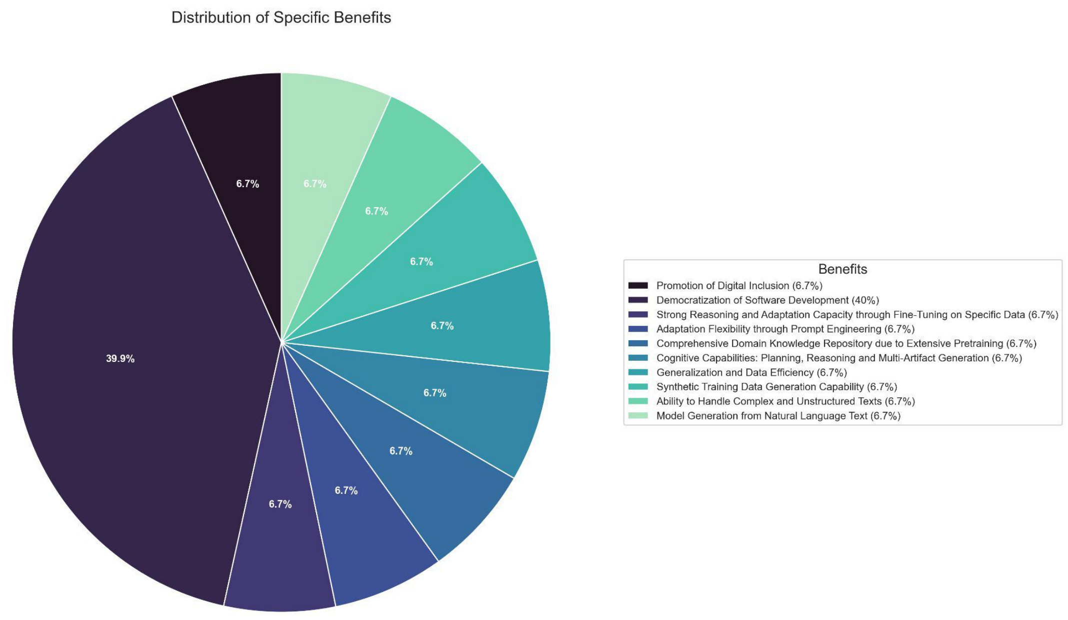

Figura 7. Distribution of specific benefits.

3.2.2. Limitations

Generative Artificial Intelligence still operates under severe technical and qualitative constraints. The greatest volume of criticism falls on the Quality and Scarcity of Data, unfortunately, the models suffer from a lack of specific datasets for Software Engineering.

The recurrence of syntactically correct but logically flawed code requires a glass ceiling for total automation. In addition, the high computational cost and low interpretability make the adoption of GenAI a high-risk decision for critical systems. We conclude that the current technology is excellent for support and prototyping, but limited for unsupervised autonomy.

3.2.3. Gaps of Research

The gaps identified show a shift in scientific interest: the focus is no longer just ”whether AI can code,”but rather how it behaves in the complete Software Lifecycle. However, there is a repressed demand for studies on the use of GenAI in Requirements, Maintenance, and integration with DevOps practices.

The science lacks long-term Empirical Metrics and Evaluations! Most studies focus on isolated tasks and not on the longitudinal impact on productivity or system quality. There is also a vast opportunity in Multi-Agent Systems and Collaboration, suggesting that the future of the field does not lie in a single giant model, but in ecosystems of specialized agents that collaborate with each other and with humans.

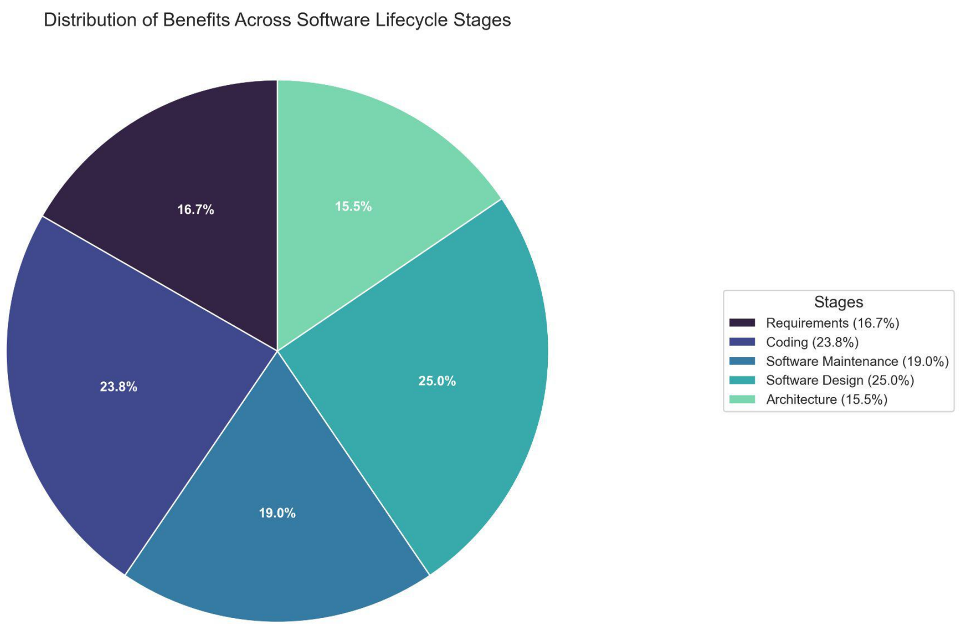

Figura 8. Distribution of Benefits Across Software Lifecycle Stages

3.2.4. Challenges

The practical barriers to these tools leaving academia and safely entering businesses encompass the Security, Privacy, and Legal Issues axis, which is the most challenging, with real concerns about intellectual property leaks and compliance with copyright laws.

In addition, Human and Organizational Factors represent the challenge of how to train developers not to over-rely on AI and how to integrate these tools into workflows that are already complex. The final challenge is Standardization: without universal benchmarks and metrics, organizations have difficulty auditing the quality of what is being delivered by GenAI-based systems.

3.2.5. Future Works reported

The analysis of the selected studies reveals six convergent thematic axes that outline key directions for future research and practice in Software Engineering with Large Language Models (LLMs).

- **Data and Experimental Infrastructure**: The studies indicate that the quality, provenance, and curation of data are critical factors for the validity of reported results. Issues such as duplicated data, data leakage, and defects in source code used for pre-training may artificially inflate model performance in controlled settings, compromising generalization to real-world scenarios. As a future research direction, there is a strong need for new public, diverse, and well-documented datasets, particularly to support activities such as requirements elicitation, in order to improve reproducibility and empirical reliability.
- **Ethics, Privacy, and Transparency**: The literature consistently highlights that the increasing adoption of LLMs requires stronger ethical frameworks. Future

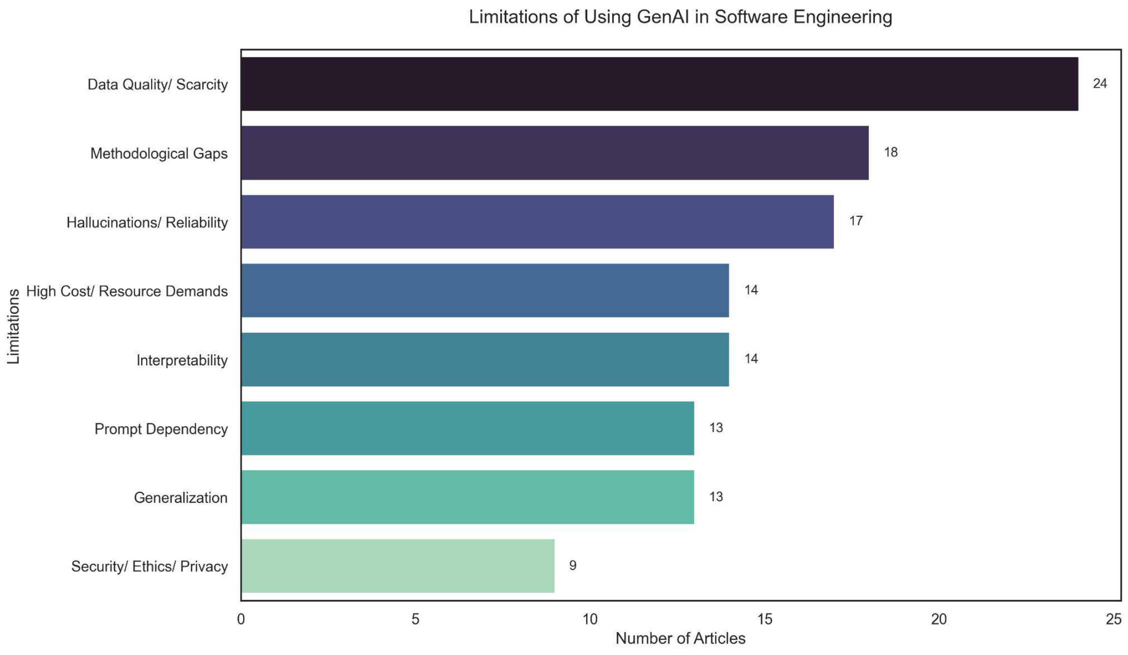

Figura 9. Limitations of using AI in Software Engeneering

work should further investigate ethical and privacy implications, especially in sensitive domains such as software security. Moreover, there is a need for methodologies, guidelines, and governance mechanisms that promote the ethical, transparent, and responsible use of generative AI, along with systematic strategies for risk mitigation.

- **Human Factors and Expert Supervision**: The findings emphasize that, despite the high automation capabilities of LLMs, the human role remains central. The interpretation of contextual nuances, code review, and assurance of accessibility, security, and overall software quality still rely heavily on human expertise. Evidence suggests that LLM-only solutions may introduce vulnerabilities or compliance issues, whereas human-in-the-loop approaches tend to maximize effectiveness, reliability, and safety.
- **Underexplored Software Engineering Areas**: Several Software Engineering domains remain insufficiently explored in the context of LLMs. These include log defect detection, the use of AI in agile software development, applications in open-source projects, and—most prominently—Requirements Engineering. In particular, there are promising opportunities for developing conversational agents capable of interviewing stakeholders, as well as for conducting studies that evaluate LLM-based solutions in real-world settings.
- **Industry Collaboration and Real-World Validation**: Another recurring theme is the gap between academic research and industrial practice. Most existing studies are conducted in controlled environments, limiting their practical applicability. Future research should prioritize academia–industry collaboration, validation in real-world projects, integration of LLM-based solutions with industrial tools, and access to real industrial data to enhance practical impact.
- **Taxonomies and Conceptual Consolidation**: Finally, there is a growing effort to propose taxonomies that organize emerging knowledge, such as prompt patterns,

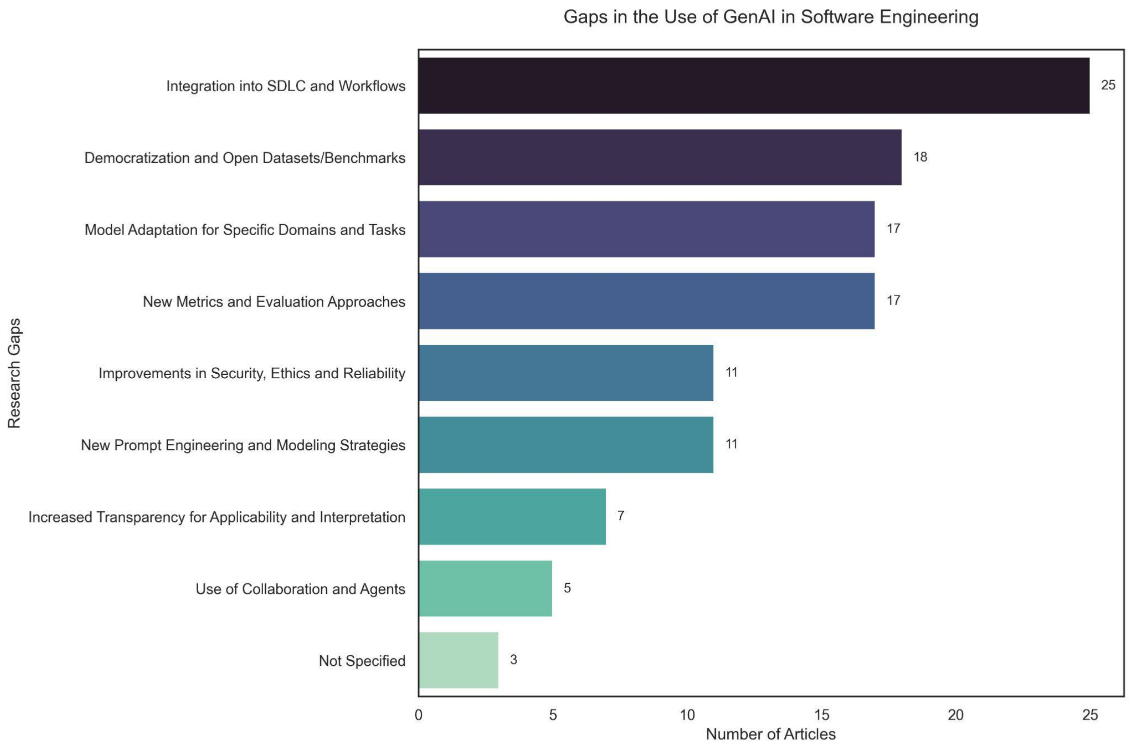

Figura 10. Gaps in the use of GenAI in Software Engeneering

the evolution of deep learning techniques for program repair, and prompt engineering practices in Software Engineering. Future work can expand, empirically validate, and integrate these taxonomies, contributing to conceptual standardization and the scientific maturity of the field.

Overall, these directions indicate that future work must address not only technical challenges but also socio-technical, ethical, and organizational aspects, reinforcing that the successful adoption of LLMs in Software Engineering depends on integrated approaches that align data, people, processes, and industrial context.

4. Threats to Validity

Reporting systematic literature studies presents intrinsic difficulties, e.g., findings are often relevant on a specific topic. We adopted the checklist presented by Ampatzglou et al. [Ampatzoglou et al. 2019] to list the actions we took to mitigate threats to the validity of this study. The tertiary study protocol was proposed by one re- searcher and reviewed by two more experienced ones. Further information about the tertiary study protocol not available in this paper can be found in the supplementary artifacts.

To mitigate search string-related threats, we selected consensual knowledge about RE, as defined in SWEBOK v4.0, and terms related to AI/ML validated through multiple pilot searches. Inclusion and exclusion criteria were discussed among the research team of this tertiary study to obtain a common understanding of gen AI concepts in SE phases. To identify relevant SLRs and mitigate search and selection biases, we searched papers through Scopus sources. Each paper was reviewed by three researchers. In case of disagreements, we resolved them through discussions and reconsiderations with the most

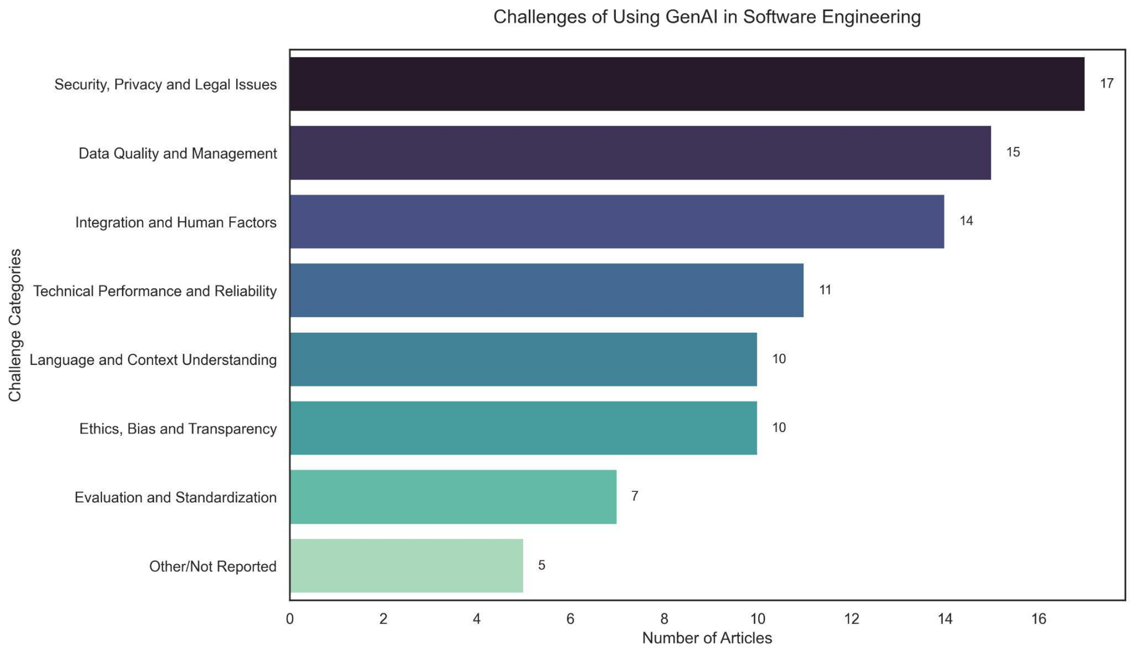

Figura 11. Challenges of using GenAI in Software Engeneering

experienced researchers. To ensure that the theoretical concepts were correctly represented during extraction and synthesis, we used the RE activities described in SWEBOK v.4.0. This decision aligns the observation of the extracted data with a reference literature.

We also defined a data extraction form to ensure consistency in extracting relevant information, and we evaluated the data according to the research questions. In addition, we had at least three researchers who extracted the data independently. Concerning the quality of the nine secondary studies found, we elaborated on quality criteria based on the CDR approach. To mitigate risks, two researchers carried out the quality evaluation process (R1 and R2), and two more experienced ones reviewed it. We also synthesized the results following the RE activities based on the SWEBOK v4.0 definitions.

Finally, despite all the effort spent to search for as many secondary studies on RE for AI/ML as possible, we are aware that some secondary research may not have been retrieved, which would restrict the generalizability of our results.

5. Final Considerations

As a next step, the creation of an ecosystem of specialized intelligent agents is proposed, designed to systematically and in a coordinated manner support the activities of Software Engineering. The approach adopts a multi-agent paradigm, in which each stage of the process is handled by an agent or a set of specialized agents, all orchestrated by a central planning agent.

The planning agent is responsible for: decomposing the overall objectives of the engineering process into executable tasks; defining the sequence and dependency between activities; coordinating the interaction between the specialized agents; consulting a common knowledge base, such as SWEBOK, to ensure conceptual, terminological, and methodological alignment between the agents.

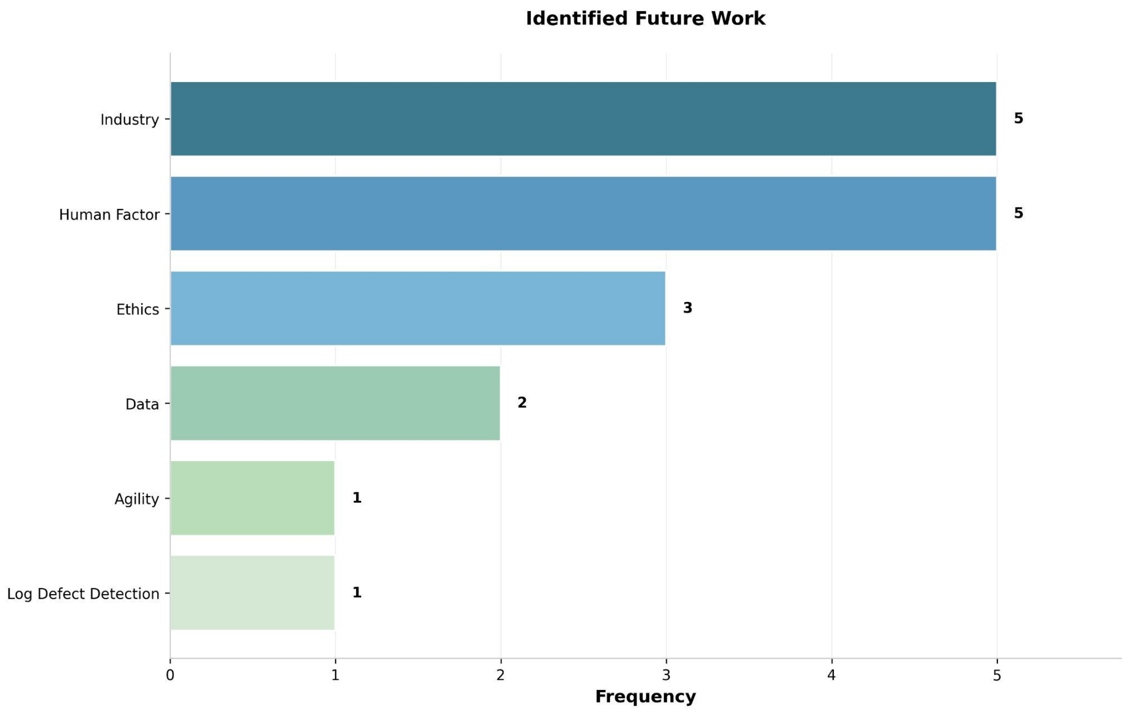

Figura 12. Identified Future Work

Each specialized sub-agent is designed to act on a specific task of the process, possessing its own objectives, context, and tools. Taking Requirements Engineering as an example, the set of agents includes:

Elicitation Agent, responsible for supporting the identification and collection of requirements from different sources (stakeholders, documents, existing systems). This agent consults relevant knowledge bases and standards, such as ISO standards, elicitation techniques, and documented best practices, generating structured requirements lists or interview and workshop records as artifacts.

Analysis Agent, responsible for examining, refining, and validating the elicited requirements, identifying ambiguities, inconsistencies, conflicts, and dependencies. This agent uses conceptual models, quality criteria, and normative guidelines as context, producing artifacts such as requirements models, dependency matrices, or validation reports.

Specification Agent, focused on the formal documentation of requirements, ensuring clarity, completeness, and traceability. It consults specific standards and templates (e.g., ISO/IEC standards for requirements specification) and generates artifacts such as structured requirements documents, user stories, or use cases.

Requirements Management Agent, responsible for tracking changes, versioning, prioritization, and traceability throughout the software lifecycle. This agent interacts with support tools and best practices, producing actions and artifacts such as change logs, traceability matrices, and impact reports.

Each specialized agent consults specific tools and knowledge bases, treated as operational context, including ISO standards, best practice guides, knowledge repositories, and project history. The interaction between agents occurs in a coordinated manner, with the planning agent ensuring global consistency, knowledge reuse, and alignment with the principles of Software Engineering.

At the end of each stage, each agent explicitly generates an artifact or performs the necessary action, allowing traceability, auditability, and continuous integration between phases. This approach enables greater automation, specialization, and control of the process, while preserving methodological coherence and alignment with established industry standards.

Referências

(2025). *Issues In Information Systems.*

(2025). How reliance on genai might limit human creativity and critical thinking in different fields.

Abdullahi, S., Usman Danyaro, K., Zakari, A., Abdul Aziz, I., Amila Wan Abdullah Zawawi, N., and Adamu, S. (2025). Time-series large language models: A systematic review of state-of-the-art. *IEEE Access*, 13:30235–30261.

Abou-Zahra, S., Brewer, J., and Cooper, M. (2018). Artificial intelligence (ai) for web accessibility: Is conformance evaluation a way forward? In *Proceedings of the 15th International Web for All Conference*, W4A '18, New York, NY, USA. Association for Computing Machinery.

Ahmad, T., Butkovic, M., and Truscan, D. (2025). Using reinforcement learning for security testing: A systematic mapping study. In *2025 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 208–216.

Ahuja, V. K., Clark, J., and WurtenBerg, J. (2025). Impact of artificial intelligence on open-source software development. *The International FLAIRS Conference Proceedings*, 38.

Al-Hashimi, H. A., Khan, R. A., Alwageed, H. S., Algarni, A. M., Ayouni, S., and Almagrabi, A. O. (2025). Exploring the role of generative ai in enhancing cybersecurity in software development life cycle. *Array*, 28:100509.

Albuquerque, B. V. L. d., Cunha, A. F. S. d., Souza, L., Siqueira, S. W. M., and Santos, R. P. d. (2024). Generating and reviewing programming codes with large language models: A systematic mapping study. In *Proceedings of the 20th Brazilian Symposium on Information Systems*, SBSI '24, New York, NY, USA. Association for Computing Machinery.

Alomari, N., Redah, M., Ashraf, A., and Alshayeb, M. (2026). Using llms to enhance code quality: A systematic literature review. *Information and Software Technology*, 190:107960.

Alturayeif, N., Hassine, J., and Ahmad, I. (2025). Machine learning approaches for automated software traceability: A systematic literature review. *Journal of Systems and Software*, 230:112536.

Ampatzoglou, A., Bibi, S., Avgeriou, P., Verbeek, M., and Chatzigeorgiou, A. (2019). Identifying, categorizing and mitigating threats to validity in software engineering secondary studies. *Information and Software Technology*, 106:201–230.

Asvydyte, G., Pandey, S. K., and Chand, S. (2025). *How Well Small Language Models Can Be Adapted for Software Maintenance and Refactoring Tasks*, page 506–515. Springer Nature Switzerland.

Barrientos, M., Winter, K., and Rinderle-Ma, S. (2025). *Automatic Extraction and Formalization of Temporal Requirements from Text: A Survey*, page 259–278. Springer Nature Switzerland.

Bouzid, R. and Khoury, R. (2025). Assessing the effectiveness of chatgpt in secure code development: A systematic literature review. *ACM Comput. Surv.*, 57(12).

Chen, C., Wang, B., and Lin, Y. (2025a). *A Systematic Mapping Study of LLM Applications in Mobile Device Research*, page 163–174. Springer Nature Singapore.

Chen, Q., Li, D., Zhao, M., Wong, W. E., and Li, H. (2025b). Learning-based automated program repair: A systematic literature review. *Complex System Modeling and Simulation*, 5(4):305–322.

Cheng, H., Husen, J. H., Lu, Y., Racharak, T., Yoshioka, N., Ubayashi, N., and Washizaki, H. (2025). Generative ai for requirements engineering: A systematic literature review. *Software: Practice and Experience*, 56(2):141–170.

Cico, O., Cico, B., and Cico, A. (2023). Ai-assisted software engineering: a tertiary study. In *2023 12th Mediterranean Conference on Embedded Computing (MECO)*, page 1–6. IEEE.

Ciubotaru, B.-I. (2025). Generative artificial intelligence and large language models: A systematic review of architectures, applications, and future directions. In *2025 25th International Conference on Control Systems and Computer Science (CSCS)*, pages 380–388.

Committee, I. C. S. P. P. et al. (2022). Swebok: Guide to the software engineering body of knowledge, 2022 version beta. *IEEE Computer Society*.

Cornide-Reyes, H., Monsalves, D., Durán, E., Silva-Aravena, F., and Morales, J. (2025). Generative artificial intelligence innbsp;agile software development processes: A literature review focused onnbsp;user experience. In *Social Computing and Social Media: 17th International Conference, SCSM 2025, Held as Part of the 27th HCI International Conference, HCII 2025, Gothenburg, Sweden, June 22–27, 2025, Proceedings, Part II*, page 228–246, Berlin, Heidelberg. Springer-Verlag.

Dehaerne, E., Dey, B., Halder, S., De Gendt, S., and Meert, W. (2022). Code generation using machine learning: A systematic review. *IEEE Access*, 10:82434–82455.

Domínguez-Isidro, S., Sánchez-García, J., Morales-Utrera, A. J., and Limón, X. (2025). *Machine Learning Techniques for Automatic Program Repair: A Systematic Literature Mapping*, page 529–543. Springer Nature Switzerland.

Ebrahim, M., Guirguis, S., and Basta, C. (2025). Enhancing software requirements engineering with language models and prompting techniques: Insights from the current research and future directions. In Zhao, J., Wang, M., and Liu, Z., editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pages 486–496, Vienna, Austria. Association for Computational Linguistics.

Edirisinghe, H. and Wickramaarachchi, D. (2024). Quality assurance for llm-generated test cases: A systematic literature review. In *2024 8th SLAAI International Conference on Artificial Intelligence (SLAAI-ICAI)*, pages 1–6.

Gorissen, S. C., Sauer, S., and Beckmann, W. G. (2024). A survey ofnbsp;natural language-based editing ofnbsp;low-code applications using large language models. In *Human-Centered Software Engineering: 10th IFIP WG 13.2 International Working Conference, HCSE 2024, Reykjavik, Iceland, July 8–10, 2024, Proceedings*, page 243–254, Berlin, Heidelberg. Springer-Verlag.

Görmez, M. K., Yılmaz, M., and Clarke, P. M. (2024). *Large Language Models for Software Engineering: A Systematic Mapping Study*, page 64–79. Springer Nature Switzerland.

Gutierrez, Y., Camacho, E., Pardo, C., and Villarreal, V. (2025). Prompts engineering challenges in software code generation. *2025 IEEE VIII Congreso Internacional en Inteligencia Ambiental, Ingenieria de Software y Salud Electronica y Movil (AmITIC)*, pages 1–8.

He, J., Treude, C., and Lo, D. (2025). Llm-based multi-agent systems for software engineering: Literature review, vision, and the road ahead. *ACM Trans. Softw. Eng. Methodol.*, 34(5).

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.

Hoisl, B., Sobernig, S., and Strembeck, M. (2017). Reusable and generic design decisions for developing uml-based domain-specific languages. *Information and Software Technology*, 92:49–74.

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. 33(8).

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

Kaur, M., Rattan, D., and Lal, M. (2025). Insight into code clone management through refactoring: a systematic literature review. *Computer Science Review*, 58:100767.

Khan, A. A., Hasan, M. T., Saari, M., Kemell, K.-K., and Rasku, J. (2025). *Why Adapt RAG for Agile? Challenges, Frameworks, and the Role of Evaluator Agent*, page 22–31. Springer Nature Switzerland.

Kumara, I., Kaplan, H., Owotogbe, J., Tamburri, D. A., and van den Heuvel, W.-J. (2025). *Large Language Models for Service-Oriented Computing (LLM4SOC): Review and Research Directions*, page 90–113. Springer Nature Switzerland.

Kwok, Y. T. C. and Adil, M. (2025). *AI and Teamwork in Agile Software Development: A Systematic Mapping Study*, page 32–40. Springer Nature Switzerland.

Li, H., Bezemer, C.-P., and Hassan, A. E. (2025). Software engineering and foundation models: Insights from industry blogs using a jury of foundation models. In *2025 IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, page 307–318. IEEE.

Li, T., Zhang, X., Wang, Y., Zhou, Q., Wang, Y., and Dong, F. (2024). Machine learning for requirements engineering (ml4re): A systematic literature review complemented by practitioners' voices from stack overflow. *Information and Software Technology*, 172:107477.

Liu, H.-C., Tsai, C.-T., and Day, M.-Y. (2024). *A Pilot Study on AI-Assisted Code Generation with Large Language Models for Software Engineering*, page 162–175. Springer Nature Singapore.

Liu, Y., Lo, S. K., Lu, Q., Zhu, L., Zhao, D., Xu, X., Harrer, S., and Whittle, J. (2025). Agent design pattern catalogue: A collection of architectural patterns for foundation model based agents. *Journal of Systems and Software*, 220:112278.

Madi, N. and Binkhonain, M. (2026). A systematic literature review on logging smell detection. *Information and Software Technology*, 190:107961.

Martins, M., Jardim, B., De Castro Neto, M., and Barriguinha, A. (2025). Talking to data: A systematic review of the rise of conversational agents for visual analytics. *IEEE Access*, 13:208902–208931.

Moenks, N., Penava, P., and Buettner, R. (2025). A systematic literature review of large language model applications in industry. *IEEE Access*, 13:160010–160033.

Muzammel, C. S., Spichkova, M., and Harland, J. (2025). Towards Using Personas in Requirements Engineering: What Has Been Changed Recently? . In *2025 IEEE 33rd International Requirements Engineering Conference Workshops (REW)*, pages 115–122, Los Alamitos, CA, USA. IEEE Computer Society.

Naidu, N. and El-Gayar, O. (2025). A review of reasoning in artificial agents using large language models. In *Proceedings of the 57th Hawaii International Conference on System Sciences*, HICSS. Hawaii International Conference on System Sciences.

Necula, S.-C., Dumitriu, F., and Greavu-Șerban, V. (2024). A systematic literature review on using natural language processing in software requirements engineering. *Electronics*, 13(11):2055.

Parikh, N. A. (2023). Empowering business transformation: The positive impact and ethical considerations of generative ai in software product management – a systematic literature review.

Pauzi, Z. and Capiluppi, A. (2024). *Beyond the Systematic: Forecasting Importance and Emergence of Research Areas in Applications of Software Traceability Using NLP*, page 119–140. Springer Nature Switzerland.

Platt, M. and Platt, D. (2023). Effectiveness of generative artificial intelligence for scientific content analysis. In *2023 IEEE 17th International Conference on Application of Information and Communication Technologies (AICT)*, pages 1–4.

Radliński, and Swacha, J. (2025). Large language models for early-stage software project estimation: A systematic mapping study. *Applied Sciences*, 15(24):13099.

Ramírez, L. C., Limón, X., Sánchez-García, J., and Pérez-Arriaga, J. C. (2024). State of the art of the security of code generated by llms: A systematic literature review. In *2024 12th International Conference in Software Engineering Research and Innovation (CONISOFT)*, pages 331–339.

Ramírez-Rueda, R., Benítez-Guerrero, E., Mezura-Godoy, C., and Bárcenas, E. (2024). A systematic literature review of 10 years of research on program synthesis and natural language processing. *Programming and Computer Software*, 50(8):725–741.

Ronanki, K., Arvidsson, S., and Axell, J. (2025). Prompt engineering guidelines fornbsp;using large language models innbsp;requirements engineering. In *Software Engineering and Advanced Applications: 51st Euromicro Conference, SEAA 2025,* *Salerno, Italy, September 10–12, 2025, Proceedings, Part III*, page 245–262, Berlin, Heidelberg. Springer-Verlag.

Rossi, B. and Fontoura, L. (2025). Ai-based approaches for software tasks effort estimation: A systematic review of methods and trends. In *Proceedings of the 27th International Conference on Enterprise Information Systems*, page 144–151. SCITEPRESS - Science and Technology Publications.

Sajadi, A., Damevski, K., and Chatterjee, P. (2026). Psycholinguistic analyses in software engineering text: A systematic mapping study. *Information and Software Technology*, 189:107913.

Santos, P. d. O., Figueiredo, A. C., Nuno Moura, P., Diirr, B., Alvim, A. C. F., and Santos, R. P. D. (2024). Impacts of the usage of generative artificial intelligence on software development process. In *Proceedings of the 20th Brazilian Symposium on Information Systems*, SBSI '24, New York, NY, USA. Association for Computing Machinery.

Sasaki, Y., Washizaki, H., Li, J., Sander, D., Yoshioka, N., and Fukazawa, Y. (2024). Systematic literature review of prompt engineering patterns in software engineering. In *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 670–675.

Sasaki, Y., Washizaki, H., Li, J., Yoshioka, N., Ubayashi, N., and Fukazawa, Y. (2025a). Landscape and taxonomy of prompt engineering patterns in software engineering. *IT Professional*, 27(1):41–49.

Sasaki, Y., Washizaki, H., Li, J., Yoshioka, N., Ubayashi, N., and Fukazawa, Y. (2025b). Landscape and taxonomy of prompt engineering patterns in software engineering. *IT Professional*, 27(1):41–49.

Shameem, M., Nadeem, M., and Niazi, M. (2024). Ai-enabled software engineer: A taxonomy of challenges and success factors (p). In *Proceedings of the 36th International Conference on Software Engineering and Knowledge Engineering*, volume 2024 of *SEKE2024*, page 69–74. KSI Research Inc.

Siddeshwar, V., Alwidian, S., and Makrehchi, M. (2024). A systematic review of ai-enabled frameworks in requirements elicitation. *IEEE Access*, 12:154310–154336.

Sohail, S. S., Farhat, F., Himeur, Y., Nadeem, M., Madsen, D. , Singh, Y., Atalla, S., and Mansoor, W. (2023). Decoding chatgpt: A taxonomy of existing research, current challenges, and possible future directions. *Journal of King Saud University - Computer and Information Sciences*, 35(8):101675.

Syahputri, I. W., Budiardjo, E. K., and Putra, P. O. H. (2025). Unlocking the potential of the prompt engineering paradigm in software engineering: A systematic literature review. *AI*, 6(9):206.

Tony, C., Díaz Ferreyra, N. E., Mutas, M., Dhif, S., and Scandariato, R. (2025). Prompting techniques for secure code generation: A systematic investigation. *ACM Trans. Softw. Eng. Methodol.*, 34(8).

Tufano, R., Pascarella, L., and Bavota, G. (2023). Automating code-related tasks through transformers: The impact of pre-training. In *Proceedings of the 45th International Conference on Software Engineering*, ICSE '23, page 2425–2437. IEEE Press.

Umama, Usman Danyaro, K., Nasser, M., Zakari, A., Abdullahi, S., Khanzada, A., Muntasir Yakubu, M., and Shoaib, S. (2025). Llm-based code generation: A systematic literature review with technical and demographic insights. *IEEE Access*, 13:194915–194939.

Vasudevan, P. and Reddivari, S. (2025). The role of generative ai models in requirements engineering: A systematic literature review. In *Proceedings of the 2025 ACM Southeast Conference*, ACMSE 2025, page 188–194, New York, NY, USA. Association for Computing Machinery.

Vitale, A., Oliveto, R., and Scalabrino, S. (2025). A catalog of data smells for coding tasks. *ACM Trans. Softw. Eng. Methodol.*, 34(4).

Zubair, F., Al-Hitmi, M., and Catal, C. (2025). The use of large language models for program repair. *Computer Standards amp; Interfaces*, 93:103951.