Use of LLMs in Requirements Engineering: An Exploratory Study in Industry

Fabio Levy Siqueira[1][0000−0002−7550−2000]1, Caio Moussa Assumpção[1], João Marcos de Mattos Barguil[2][0000−0001−9302−8683], and Julio Cesar Sampaio do Prado Leite[1][0000−0002−0355−0265]

[1] Universidade de São Paulo, São Paulo, SP, Brazil
{levy.siqueira,caioma,julioleite}@usp.br
[2] Opus Software,
jbarguil@opus-software.com.br

Abstract. Large language models (LLMs) can support requirements engineering activities in different ways. Prior studies have investigated this potential through surveys and interviews with practitioners, focusing on how LLMs are reportedly used in RE tasks. However, these studies do not examine the actual prompts employed by software engineers in real-world contexts. To address this gap, this exploratory study analyzes a log of interactions between software engineers and LLMs during the execution of requirements engineering activities. These interactions are compared with the most frequently reported uses of LLMs in requirements engineering, as identified through queries to Gemini, ChatGPT, and Claude. Our motivation is to gain insights into how practitioners are using LLMs in their requirements engineering practice. The results indicate that software engineers issue a wide variety of requests to LLMs, many of which diverge from the commonly reported usage patterns. Furthermore, the identified queries span all requirements engineering activities, suggesting a broader and more heterogeneous use of LLMs.

Keywords: LLM · requirements engineer · chat.

1 Introduction

In recent years, there has been a growing popularization of the use of large language models (LLMs), such as ChatGPT, Claude, and Gemini. According to a 2025 McKinsey survey [3], 88% of the respondents' organizations report using AI for at least one business function, with the use of AI agents being more prevalent in technology companies, particularly for software engineering.

Specifically in requirements engineering (RE), AI can support all its parts [4] - elicitation, modeling, analysis, and management. LLMs can be used to analyze existing documentation, format requirements into specific representations and templates, categorize requirements, assess requirement quality, and simulate stakeholder perspectives, among other applications [4]. Despite these possibilities, it remains unclear how requirements engineers actually use LLMs

Proceedings of the 29th Workshop on Requirements Engineering (WER2026),
August 11-13, 2026, La Plata, Argentina.
https://doi.org/10.29327/1884517.29-17

in practice. Our motivation is to gain knowledge on how practitioners are using LLMs in their requirements engineering practice.

The analyzed chats were obtained from the real-world use of LLMs by software engineers in their professional activities. We had access to a log of interactions between software engineers and LLMs from Opus Software, a custom software shop with 35 years of experience, serving clients ranging from large corporations to early-stage startups. The chats[3] reflect real-world use of LLMs by professionals in their activities, including requirements engineering tasks. With this data, one of the questions that came to us was to which extended these interactions were similar to frequent LLMs interaction of other professionals. Knowing more about LLM usage, in a particular setting, would add to the knowledge of how practitioners do use LLM for RE tasks.

As such, we conducted a review of the literature. We have identified three related literature reviews [6,7,18]. In their reviewers we did not find a reference to studies on how practitioners were using LLMs for requirements, on the contrary. Zadenoori et al [18, p.45] report the following: "This is confirmed by the limited use of Field Studies and Experimental Simulations, which suggests a gap in understanding real-world applicability.", and "indicates that we currently have no evidence of the viewpoint of practitioners across different companies on LLM4RE."

The systematic review conducted by [6] is centered on research on LLMs' role in RE, but not research on their use by practitioners. The authors mention: "The following are real-world prompt engineering examples applied in SRE." However, the examples are from experiments with available data [12,16] , but not from observed/real use by practitioners.

In [7] the authors explicitly observe the lack of strategies for the use of LLMs by practitioners, concluding that: "This approach will help accelerate the adoption of LLM-driven methods in real-world SE scenarios." [7, p.556], so it also does not deal with the use of LLM by practitioners.

Therefore, the objective of this work is to analyze how practitioners use LLMs in their RE activities. To this end, we present an exploratory study that examines chats between software engineers and LLMs from a company, Opus Software, when performing requirements engineering tasks, comparing these interactions with the most frequent uses of LLMs in RE. A total of 102 chats between April 2024 and April 2025 were considered, involving different versions of ChatGPT and Gemini.

The identification of the most frequent uses of LLMs was carried out by directly querying two LLMs, with a third LLM used to summarize the resulting information. The conversations were then mapped to these uses by a requirements engineering specialist.

This paper is organized as follows. Section 2 presents the background and related works, discussing large language models and the use of LLMs in RE. Section 3 describes the method used to identify the most frequent queries and

[3] For confidentiality reasons, it is not possible to disclose the content of these chats, as they involve information from commercial projects.

to map the chats to them. Section 4 presents the results of the chat analysis, including the context and a discussion of the findings. Finally, Section 5 presents the conclusions and future works.

2 Background and Related Work

Requirements engineering, like other areas of software engineering, can benefit from the use of LLMs [1]. Next we discuss the concept of large language model and its use in requirements engineering, discussing related works.

2.1 Large Language Models (LLM)

According to [2, p.4], language models are "Artificial Intelligence models specialized in understanding, processing and generating human (or natural) language." Large Language Models (LLMs) are language models pre-trained on huge amounts of data.

Several techniques have been proposed for modeling natural language, including bag-of-words and word2vec. However, the Transformer architecture [2] has become the predominant due to its ability to represent inputs more accurately and faster, as well as the possibility for parallel training, which significantly accelerates the training process.

The original Transformer architecture use stacked encoders and decoders. Encoders are components that represent natural language, while decoders generate it. Subsequent models have different configurations. For example, BERT is an encoder-only architecture, while generative models, such as those in the GPT family, rely exclusively on decoder components [2].

Currently, generative models, such as ChatGPT, Gemini, and Claude, are widely used. These models can perform a large variety of tasks with no additional training, as well as having text as input, making their use straightforward. However, generating prompts is not trivial nor exact, so some techniques and heuristics have been designed to write a prompt that guide the machine to a desired output [5].

2.2 Use of LLMs in Requirements Engineering

Some studies suggest potential uses of LLMs for RE [1,4]. However, to understand how requirements engineers are actually using LLMs in practice, it is important to rely on studies based on data from real users. In the quest for primary studies with such data, we used three strings in Google Scholar: 1) "use of LLM for requirements engineering report from practitioners," 2) "LLM usage for requirements by professionals or practitioners interview survey," and 3) "LLM use requirements professionals frequent queries."

We looked at the first 20 answers for each query, and selected the ones which could be of interest, using the snippet produced by Google Scholar. We looked at 12 papers, and read 6 of them, which we believed were more akin to our goal.

The work of Santos et al. [13] aims to explore how requirements engineers perform "elicitation and validation" and how they use LLMs, with three research questions. One is about daily RE professional activities, another how professionals use LLM in their RE activities, and the last about the RE professionals' perception on user stories generated by LLMs. They gathered responses from 21 professionals. Although respondents enumerated their choice of elicitation and validation strategies, the LLM usage queries was just for generation of user stories. Using the available open data of [13] we found that 8 respondents used LLMs, but only 6 used it for user story generation, and two used for "Code generation, Initial ideas for brainstorming, Technical documentation writing, Requirements analysis or refinement, Initial ideas for brainstorming." The work did not ask which prompts were used. In the work the respondents reported on what they use it for, but without detail on how (prompt used) they interact with the LLM.

The work of Pasquale et al. [9] reports on the use of three LLMs (GPT3, GPT4, and Llama) in a joint work with an IT consulting company. In this case, the LLMs performed the tasks of summarization of three textual documents, user story generation from the summary, the epic Functional Design Specification (FDS) Generation from the summary, and the merging of these two specifications into a FDS. The paper provides open data with the prompts used in joint work. The LLMs results were compared with results produced by requirements engineers with positive results, but with a caveat: "LLMs can help automate and standardize the requirements specification, reducing time and human effort. However, the quality of LLM-generated FDS highly depends on input and often requires human revision." [9, p.389]. Although the work did not report on actual day-to-day use, it brings up categories of use, which the joint task force found in need, if LLMs were to be adopted in the company.

Rani et al. [11] tackles the adoption of AI in RE practices. A survey with 55 software practitioners, with different roles, was conducted. The paper reports on: Adoption Levels, AI integration levels, and Responsible AI Practices. In the case of adoption, specification and validation activities are prevalent over elicitation and analysis. About the AI integration levels, which shows if decisions rely more on human judgment, or more on AI, and to which side the collaboration is towards human or AI (HAIC), the activities elicitation and validation are the one where human judgement is preponderant. Regarding the Responsible AI Practices, the results confirm that the respondents believe that the human must be in the loop. When asked about the relation of usage categories and activities more practitioners related elicitation with the category "Can automate significant parts but requires human oversight." Although the paper does not report on the details of how practitioners do interact with AI, they provide data [10] on adoption levels, on the balance of decision making in using AI, and categories of use identified in the survey.

Spijkman et al. paper [15] adopted a process-oriented study in the context of a low-code consultancy company using Agile Model-Driven Development. Within three company projects, they interviewed practitioners in the roles of business

analyst or lead developer. The interview was used to elicit the case of the projects within the MDD process and to probe the practitioners on how they would envision using LLMs in their workflow. Based on the practitioners' responses [14], the authors identified the main 5 categories in which LLMs would be used: User Story, Generation Acceptance Criteria Generation, Data Model Generation, Process Model Generation, and Meeting Summarization. With these categories and using previously published LLM prompt patterns for requirements, they built the prompts, executed them on project data and evaluated them. This paper does not delve into how practitioners performed in their day-to-day activities, however the data related on practitioner's look ahead in LLM usage is of interest to compare with real usage.

Although the papers are not directly comparable to our work, they are related in several aspects, as we have noticed. The fact that they provide open data, it also brings an opportunity for future work on relating our findings with the data they provide.

3 Method

The method, presented in Figure 1, comprises three components: (1) the use of the IRES framework [8] to anchor our motivation, by eliciting the context; (2) the use of intentional modeling to explain the strategy for deriving a set of the 50 most frequent queries; and (3) the use of a mapping strategy to relate chats to these 50 most frequent queries.

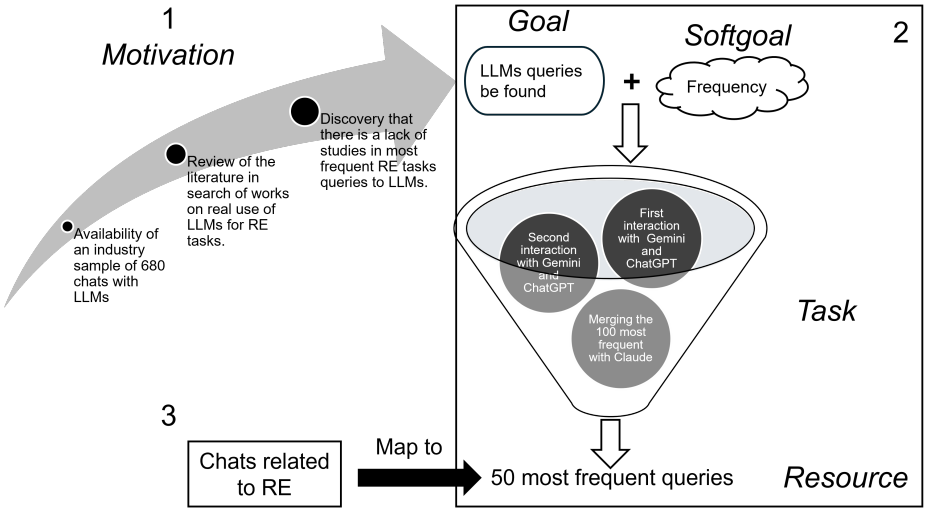

Fig. 1. Research method comprising of three components.

The first component is inspired by the IRES framework [8]. It is described by the elicitation of context information that grounded our motivation, as seen on the arrow on the top-left part of Figure 1.

The second component, on the right part of Figure 1, is based on the i* framework [17]. Our motivation was to understand the actual use of LLMs in requirements engineering tasks. Our goal and softgoal were ask LLMs. As the first task, we asked Gemini and ChatGPT to provide queries/prompts they have received from software developers related to requirements engineering responsibilities and to categorize them into the following categories: elicitation, modeling, analysis (verification and validation), and management. The second task consisted of having ChatGPT and Gemini answer the 50 most frequent queries/prompts from software engineers with responsibilities in the area of requirements engineering[4].

The answers were credible, and at the end, we obtained the 100 most frequent queries received by these LLMs. However, for our initial work on the most frequent queries, we needed to summarize these 100 queries into the top 50 most frequent ones.

To achieve this, we interacted again with an LLM, this time using Claude. We formulated a structured prompt to generate a consolidated list of 50 queries, ordered by frequency (from the most to the least frequent), considering the persona of a requirements engineering researcher. The result is presented in Table 1.

The third component of the method, presented in the lower-left part of the Figure 1 consists of mapping real software engineers' chats related to RE to the 50 queries. Each chat corresponds to an interaction session between the LLM and an engineer and, therefore, contains multiple queries and statements posed by the engineer, as well as responses generated by the LLM. The messages (queries, statements, and responses) may include text, images, or documents.

Each chat was read carefully by a RE specialist in its entirety, considering all its queries, statements, and responses. As a result, some chats were mapped to more than one of the 50 most frequent queries. For instance, one chat contained 10 interactions between the software engineer and the LLM and was mapped to 5 of the most frequent queries. However, each chat was mapped only once to a given query, even if the same request was made multiple times within the chat.

Since some of the most frequent queries were too specific, it was necessary to abstract them for mapping purposes. For instance, in query 11, instead of considering the IEEE 830 standard, any standard was considered. Similarly, in query 33 instead of "best practices," the analysis considered the specific practices requested, such as feasibility.

4 Analysis of LLM interactions

The chats were obtained from interactions involving software engineers who perform RE activities, among other responsibilities, at Opus Software. The company

[4] The questions and answers were originally written in Portuguese and were translated into English with the assistance of an AI for presentation in this paper.

Table 1. List of 50 most frequent queries, ordered by frequency, reflecting the perspective of a requirements engineering researcher, based on responses from Gemini and ChatGPT and consolidated by Claude.

#	Query
1	Who are the main stakeholders for a system in the [domain]?
2	What questions should I ask the client/user to understand the problem and elicit requirements?
3	What are the functional requirements of a system in the [domain]?
4	What are the critical non-functional requirements for a system in the [domain]?
5	Identify user needs and potential requirements for [system/domain].
6	Transform this idea/text into clear and testable requirements.
7	Generate requirement hypotheses based on this scenario.
8	What are the existing business processes and constraints in this domain?
9	Analyze this document/transcript and extract implicit business needs.
10	Help define the scope of this project to avoid scope creep.
11	Generate a complete Software Requirements Specification (SRS) following the IEEE 830 standard for [system].
12	Write User Stories with Acceptance Criteria for [functionality].
13	Transform these informal requirements into Gherkin statements (Given / When / Then).
14	Convert this descriptive text into a structured Use Case (main and alternative flows).
15	Rewrite these requirements to remove ambiguity and make them clear and testable.
16	Generate a table of Functional Requirements based on this meeting transcript.
17	Classify and organize requirements by priority / backlog / epics and user stories.
18	Document requirements in plain language for non-technical stakeholders.
19	How can I elicit privacy/security requirements for a system handling sensitive data?
20	What are the business requirements for [system/domain]?
21	Create use cases and describe main and alternative flows for system [X].
22	Model a sequence diagram for the [functionality] process.
23	Model entities and relationships / a class diagram for the [module].
24	Identify dependencies and conflicts between requirements.
25	Identify gaps and suggest improvements to these requirements.
26	Describe the user journey for [persona] performing [task].
27	Suggest a functional decomposition structure for the [domain] system.
28	Trace requirements to business objectives.
29	Propose an architecture and map requirements to system components.
30	Evaluate architectural trade-offs based on these requirements.
31	Are these requirements complete, consistent, and testable?
32	Identify ambiguities and vague terms (e.g., “fast”, “intuitive”) in these requirements.
33	Validate these requirements based on requirements engineering best practices.
34	Suggest quality metrics and checklists to validate these requirements.
35	Generate test cases and scenarios (happy path and edge cases) from the requirements.
36	Generate acceptance criteria and tests (BDD/Gherkin) for these requirements.
37	Create a Requirements Traceability Matrix (RTM) linking requirements to test cases.
38	Classify requirements by risk and assess the impact of changes.
39	How can this non-functional requirement be made measurable and testable?
40	Which compliance requirements (e.g., LGPD, GDPR, WCAG) apply to this system?
41	Identify conflicts between non-functional requirements (e.g., security vs. usability) in this scenario.
42	Create a MoSCoW prioritization matrix for these requirements.
43	Suggest scalability and availability requirements for a high-demand system.
44	How should explainability, bias, and ethical requirements be specified for an AI system?
45	How should Human-in-the-Loop requirements be documented for this automated process?
46	Specify monitoring requirements for an AI model in production (e.g., data drift).
47	What data requirements are needed to train/operate an AI model of type [type]?
48	Create requirements addressing hallucinations and latency constraints in LLM-based interfaces.
49	How can requirements changes be managed in an agile environment while preserving documentation?
50	Act as a senior requirements engineer and review this document, identifying logical flaws.

focuses on Open Finance, digital transformation, AI, big data, and microservices. Beyond technical delivery, Opus brings applied business knowledge to its projects, an unusual combination in the industry. To keep corporate data off free AI tools, Opus built an internal AI chat platform called Opus Boost, which is

the data source for this study. It allows corporate users to interact with different LLMs, such as ChatGPT and Gemini.

The exploratory analysis considered 102 chats, randomly selected from a set of 680 chats (15%) produced by 12 software engineers.

4.1 Analysis

The 102 chats were initially classified by one of the researchers into two groups: (1) related to RE and (2) not related to RE. Chats for which the researcher was uncertain about the classification (27 cases) were subsequently reviewed by a second researcher with expertise in requirements engineering. After this review, a final set of 29 chats was obtained.

The 29 chats were analyzed by a researcher with experience in requirements engineering and mapped to the most frequent queries. During this analysis, seven chats were considered not related to requirements engineering activities. Therefore, the mapping was based on 22 chats. The results are presented in Table 2.

Table 2. Mapping of chats to the most frequent queries and RE parts.

#	Most frequent query	# of chats	RE part
33	Validate these requirements based on requirements engineering best practices.	2	Analysis
12	Write User Stories with Acceptance Criteria for [functionality].	2	Modeling
1	Who are the main stakeholders for a system in the [domain]?	1	Elicitation
2	What questions should I ask the client/user to understand the problem and elicit requirements?	1	Elicitation
3	What are the functional requirements of a system in the [domain]?	1	Elicitation
5	Identify user needs and potential requirements for [system/domain].	1	Elicitation
6	Transform this idea/text into clear and testable requirements.	1	Elicitation
7	Generate requirement hypotheses based on this scenario.	1	Modeling
11	Generate a complete Software Requirements Specification (SRS) following the IEEE 830 standard for [system].	1	Modeling
36	Generate acceptance criteria and tests (BDD/Gherkin) for these requirements.	1	Modeling

The most common uses (2 chats each) were the writing of user stories (as the company adopts the user story technique in its projects) and the validation of requirements based on best practices. All other uses appeared in only one chat each.

Not all chat queries posed are included in the list of most frequent queries. Table 3 presents these queries not identified as most frequent, indicating how many chats included them. Among these non frequent queries, the most common use was the summarization of texts related to requirements, such as transcripts of elicitation meeting. Another frequent use was explaining concepts related to requirements engineering techniques. The LLM was also asked by the software engineers about the activities required to perform a requirements engineering technique, as well as the estimated effort for such activities. Another identified use was the creation of context diagrams.

#	Non frequent queries	# of chats	RE part
53	Summarize a text related to requirements (e.g., an elicitation meeting transcript).		General
61	Explain concept [X] within the RE technique [Y].	3	Conceptual
51	What activities are involved in performing the RE technique [X]?	2	Management
52	What is the estimated effort for this RE activity?	2	Management
57	Create a context diagram.	2	Modeling
55	Who are the personas represented in these user stories?	1	Analysis
56	Compare these requirements with those of a competing system [X].	1	Analysis
63	Improve a text describing a functionality.	1	Analysis
64	Which business rules are associated with this functionality?	1	Analysis
62	Generate UI prototypes based on these functionalities.	1	Analysis
59	How can tool [X] be used to support the RE activity [Y]?	1	Conceptual
54	Create a presentation based on these requirements.	1	General
58	Generate a domain glossary.	1	Modeling
60	Break down the following epics into user stories.	1	Modeling

Table 3. Mapping of chats to non frequent queries and RE parts.

4.2 Discussion

The analysis of the chats indicates that software engineers made a wide variety of requests to LLMs, most of which were not among the most common queries identified. Considering the RE parts elicitation, modeling, analysis, and management (presented in Table 2 and 3), the most frequent use was to support requirements modeling, particularly through the generation of user stories from elicited functionality and context diagram. Another important use was related to analysis, helping software engineers better understand the defined requirements. Interestingly, these correspond to the two RE parts identified in [11] as having the greatest potential for LLMs to collaborate with humans.

In elicitation, all identified queries were mapped to those suggested by the LLMs, indicating that the professionals' usage aligned with the expected uses. Most queries focused on suggesting requirements, but there were also queries regarding stakeholders and which questions should be asked to clients in order to elicit requirements.

Two distinct types of use were general-purpose and conceptual uses of LLMs. General-purpose use included tasks such as generating summaries and presentations. Regarding conceptual queries, one possible explanation is that most of them were posed by a single software engineer who, based on the chats, appeared to have less experience. Nevertheless, the results indicate that LLMs can support requirements engineering activities in multiple ways. Concerning use for Management, it relates to managing requirements-related activities rather than managing artifacts or contextual information.

Threats to validity This work presents some threats to validity. The identified most frequent queries, obtained using three LLMs, do not necessarily represent the most popular queries. The adopted method may introduce biases, whether due to the specific queries posed, the order in which the LLMs were used, or assumptions from the models themselves. Nevertheless, the use of three LLMs was intended to mitigate such influences.

Another threat concerns the generalization of the results. The study considered chats from only one company, and these interactions reflect its organizational culture, practices, and client demands. In addition, only 22 chats related to requirements engineering were analyzed, out of 102 interactions between software engineers and LLMs, randomly selected from a larger dataset. It is also unclear to what extent these conversations were actually used to support real-world activities.

A further threat relates to the mapping process. The classification of which chats were related to requirements engineering was performed by two authors; however, the mapping itself was conducted by a single researcher. This decision was motivated by confidentiality constraints associated with the conversations.

Finally, although this an exploratory study, a concern is the limited coverage of literature, and some refereed work is, still, only available without a peer review.

5 Conclusion

This exploratory study advances the understanding of how practitioners use LLMs in their requirements engineering practice. We obtained interactions from software engineers working on real-world projects and, from a total of 102 chats, classified 22 as related to requirements engineering activities. These chats were then mapped to the 50 most frequent questions posed to LLMs for requirements engineering activities. These questions were identified using Claude to consolidate responses generated by ChatGPT and Gemini.

The results indicate that the identified uses span multiple requirements engineering activities, while also revealing the presence of conceptual and general-purpose queries. The most common uses were related to modeling and analysis.

As future work, we intend to analyze the full set of 680 interactions from a requirements engineering perspective. A key challenge lies in defining an appropriate analysis approach, particularly for classifying these interactions. To address this, we plan to leverage LLMs, while carefully considering confidentiality constraints. We also intend to explore existing open datasets to perform an additional mapping, beyond that based on the most frequent questions.

Acknowledgement

This work was supported in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) under Grant 001. Leite was partially supported by CNPq.

References

1. Ahmed, I., Aleti, A., Cai, H., Chatzigeorgiou, A., He, P., Hu, X., Pezzè, M., Poshyvanyk, D., Xia, X.: Artificial Intelligence for Software Engineering: The Journey So Far and the Road Ahead. ACM Trans. Softw. Eng. Methodol. **34**(5), 119:1–119:27 (May 2025). https://doi.org/10.1145/3719006

2. Alammar, J., Grootendorst, M.: Hands-On Large Language Models. O'Reilly Media, 1 edn. (2024)
3. Alex Singla, Alexander Sukharevsky, Bryce Hall, Lareina Yee, Michael Chui, Tara Balakrishnan: The state of AI in 2025: Agents, innovation, and transformation. Tech. rep., McKinsey & Company (Nov 2025)
4. Arora, C., Grundy, J., Abdelrazek, M.: Advancing Requirements Engineering Through Generative AI: Assessing the Role of LLMs. In: Nguyen-Duc, A., Abrahamsson, P., Khomh, F. (eds.) Generative AI for Effective Software Development, pp. 129–148. Springer Nature Switzerland, Cham (2024). https://doi.org/10.1007/978-3-031-55642-5_6
5. Chen, B., Zhang, Z., Langrené, N., Zhu, S.: Unleashing the potential of prompt engineering for large language models. Patterns **6**(6), 101260 (Jun 2025). https://doi.org/10.1016/j.patter.2025.101260
6. Hemmat, A., Sharbaf, M., Kolahdouz-Rahimi, S., Lano, K., Tehrani, S.Y.: Research directions for using LLM in software requirement engineering: a systematic review. Frontiers in Computer Science **7** (Mar 2025). https://doi.org/10.3389/fcomp.2025.1519437
7. Huang, K., Wang, F., Huang, Y., Arora, C.: Prompt Engineering for Requirements Engineering: A Literature Review and Roadmap. In: 2025 IEEE 33rd International Requirements Engineering Conference Workshops (REW). pp. 548–557 (Sep 2025). https://doi.org/10.1109/REW66121.2025.00081, iSSN: 2770-6834
8. Oliveira, A.d.P.A., Marcio Cysneiros, L., Cesar Sampaio do Prado Leite, J.: Goal elicitation heuristics anchored on a thinking frame. In: 2022 IEEE 30th International Requirements Engineering Conference Workshops (REW). pp. 195–199 (2022). https://doi.org/10.1109/REW56159.2022.00044
9. Pasquale, L., Ragone, A., Piemontese, E., Darban, A.A.: Exploring the Use of LLMs for Requirements Specification in an IT Consulting Company. In: 2025 IEEE 33rd International Requirements Engineering Conference (RE). pp. 389–399 (Sep 2025). https://doi.org/10.1109/RE63999.2025.00045, iSSN: 2332-6441
10. Rani, L.M.: Supplementary material to "ai for requirements engineering: Industry adoption and practitioner perspectives" (Jun 2025). https://doi.org/10.5281/zenodo.16926585
11. Rani, L.M., Svensson, R.B., Feldt, R.: AI for Requirements Engineering: Industry Adoption and Practitioner Perspectives. In: 2025 40th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW). pp. 244–251 (Nov 2025). https://doi.org/10.1109/ASEW67777.2025.00053, iSSN: 2151-0849
12. Ronanki, K., Cabrero-Daniel, B., Horkoff, J., Berger, C.: Requirements Engineering Using Generative AI: Prompts and Prompting Patterns, pp. 109–127. Springer Nature Switzerland, Cham (2024). https://doi.org/10.1007/978-3-031-55642-5_5
13. Santos, R., Steinmacher, I., Conte, T., Oran, A.C., Gadelha, B.: Adoption of LLMs in Requirements Engineering: What Practitioners Are Worried About? In: Simpósio Brasileiro de Qualidade de Software (SBQS). pp. 248–258. SBC (Nov 2025). https://doi.org/10.5753/sbqs.2025.15089
14. Spijkman, T., Dalpiaz, F., Overbeek, S., Bente, M., Steffen, B.: Survey results and example mock-case for: Llm- assisted requirements engineering in agile mdd: Industry insights and validation (Jun 2025). https://doi.org/10.5281/zenodo.15754874
15. Spijkman, T., Molenkamp, B., Beudeker, S., Overbeek, S., Dalpiaz, F.: LLM-Assisted Requirements Engineering in Agile MDD: Industry Insights and Validation. In: 2025 IEEE 33rd International Requirements Engineering Conference (RE). pp. 366–377 (Sep 2025). https://doi.org/10.1109/RE63999.2025.00043, iSSN: 2332-6441

16. White, J., Hays, S., Fu, Q., Spencer-Smith, J., Schmidt, D.C.: ChatGPT Prompt Patterns for Improving Code Quality, Refactoring, Requirements Elicitation, and Software Design, pp. 71–108. Springer Nature Switzerland, Cham (2024). https://doi.org/10.1007/978-3-031-55642-5_4
17. Yu, E.: Modelling Strategic Relationships for Process Reengineering. Ph.D. thesis, Graduate Department of Computer Science, University of Toronto (1995)
18. Zadenoori, M.A., Dąbrowski, J., Alhoshan, W., Zhao, L., Ferrari, A.: Large Language Models (LLMs) for Requirements Engineering (RE): A Systematic Literature Review (Sep 2025). https://doi.org/10.48550/arXiv.2509.11446