Enhancing Software Requirements Engineering with Language Models and Prompting Techniques: Insights from the Current Research and Future Directions

Abstract

Large Language Models (LLMs) offer transformative potential for Software Requirements Engineering (SRE), yet critical challenges, including domain ignorance, hallucinations, and high computational costs, hinder their adoption. This paper proposes a conceptual framework that integrates Small Language Models (SLMs) and Knowledge-Augmented LMs (KALMs) with LangChain to address these limitations systematically. Our approach combines: (1) SLMs for efficient, locally deployable requirements processing, (2) KALMs enhanced with Retrieval-Augmented Generation (RAG) to mitigate domain-specific gaps, and (3) LangChain for structured, secure workflow orchestration. We identify and categorize six technical challenges and two research gaps through a systematic review of LLM applications in SRE. To guide practitioners, we distill evidence-based prompt engineering guidelines (Context, Language, Examples, Keywords) and propose prompting strategies (e.g., Chain-of-Verification) to improve output reliability. The paper establishes a theoretical foundation for scalable, trustworthy AI-assisted SRE and outlines future directions, including domain-specific prompt templates and hybrid validation pipelines.

1 Introduction

Incomplete or ambiguous requirements result in 28% of software defects as per (Mogyorodi [43]). In today's rapidly evolving software landscape, where development cycles are compressed and business needs change constantly, this requirements gap poses significant risks to project success and competitiveness (Umar and Lano [54]). Effective requirements engineering serves as the critical foundation for software quality, with Business Analysts playing a pivotal role in bridging the stakeholder needs and their technical implementation (Wiegers and Beatty [59]). Software Requirements Engineering (SRE) systematically transforms stakeholder inputs into complete and consistent specifications through elicitation, analysis, specification, validation, and management (Project Management Institute (PMI) [46]), (International Institute of Business Analysis (IIBA) [28]). However, the natural language nature of requirements introduces challenges in precision and scalability that traditional methods struggle to address. These challenges can now be addressed by the evolution of Large Language Models (LLMs), which leverage advanced NLP techniques to automate requirements engineering tasks.

Large Language Models (LLMs) present a transformative opportunity for SRE. Their advanced natural language capabilities enable automation of requirements elicitation (Hey et al. [23]), ambiguity detection (Sainani et al. [50]), and specification generation (Dalpiaz and Niu [10]). Practical applications like GitHub Copilot (Ronanki et al. [48]) and ChatGPT-4 (Brown et al. [6]) demonstrate their potential in understanding linguistic context and stakeholder intent (Kaur et al. [30], Winkler and Vogelsang [60]). LLMs can simulate user roles (Wei [56]), analyze requirement quality (Ferrari et al. [18]), and even suggest improvements (Luo et al. [35], Alhoshan et al. [1]).

However, LLM adoption faces significant challenges. Output quality concerns include potential inaccuracies, biases, and lack of transparency (Marques et al. [38], Zhen et al. [62]). The effective utilization of LLMs requires sophisticated prompt engineering techniques (Sahoo et al. [49]) that understand model behavior and task requirements (Fan et al. [17]). Current research provides frameworks for prompt design (Liu and Chilton [34], Hao et al. [21], Maddigan and Susnjak [36]) and commercial implementations (OpenAI [44]), with emerging applications specifically for requirements engineering (Bang et al. [3], Arora et al. [2]).

This paper investigates the application of Large Language Models (LLMs) in Software Require-

ments Engineering (SRE), analyzing current technical and methodological challenges while projecting future directions for LM integration. Building upon foundational survey research in LLMs and prompt engineering, we systematically synthesize existing knowledge to: (1) identify key challenges in LLM-SRE adoption, (2) propose a conceptual framework for addressing these challenges, and (3) establish evidence-based prompting guidelines for requirements engineering tasks. While this study establishes a theoretical foundation for integrating LLMs into SRE workflows, the technical implementation and empirical validation remain important directions for future research. Our work provides a structured framework to bridge the critical gap between cutting-edge language model capabilities and rigorous requirements engineering practices, offering reproducible methodologies for both researchers and practitioners.

2 Background and related works

2.1 Software Requirements Engineering

Software requirements define the framework and primary objectives that guide the development of a software application (International Institute of Business Analysis (IIBA) [28]). The process of crafting, documenting, and managing these requirements is known as requirements engineering (Bencheikh and Höglund [4]). As a disciplined and structured approach, software requirements engineering focuses on consistently defining, documenting, and maintaining requirements throughout the software development life cycle (Wiegers and Beatty [59]). SRE can be decomposed into 2 main areas, which are requirements development and requirement management (Marques et al. [38]) (Westfall [58]). The development involves requirements elicitation, analysis, and specifications, while management is a continuous process over the development life cycle that covers change requests, documents, and tracing the history of the requirement.

Since software requirements are being written and communicated in a natural language, this drove the extensive research on the usage of NLP techniques and approaches in the SRE field (Dalpiaz et al. [9]). A common approach for supporting RE tasks would be the usage of Language Models to facilitate the management of various RE activities by reducing time consumption, complexity, and human effort (Kaur et al. [30]), (Winkler and Vogelsang [60]). NLP, powered by AI and computational techniques, enables interaction between AI systems and humans in natural language, enhancing the efficiency of these tasks. However, for large language models (LLMs) to be effectively applied within RE, they must gain a contextual understanding of RE activities and acquire domain-specific knowledge.

2.2 Language Models

Language Models (LMs) trace their origins to early efforts in natural language processing (NLP), but it wasn't until the emergence of neural networks and deep learning that LLMs began to gain Significance. Early developments like Word2Vec (Mikolov et al. [41]) laid the groundwork by allowing models to learn word representations from large datasets. The real breakthrough came with the introduction of the transformer architecture by Vaswani et al. in their 2017 paper Attention is All You Need (Vaswani et al. [55]). This innovation allowed models to handle context more effectively and perform tasks such as translation, summarization, and question answering with higher accuracy.

The evolution of Language Models was accelerated by the development of larger models trained on massive datasets. OpenAI's GPT (Generative Pre-trained Transformer) series, particularly GPT-3, showcased how scaling model size and training on diverse textual corpora could enable models to perform a wide range of tasks without task-specific training (Brown et al. [6]). Similarly, BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al. [13]) revolutionized contextual understanding by processing text bidirectionally. This evolution reflects a shift from task-specific to general-purpose models capable of handling various NLP tasks. The introduction of Meta's LLAMA (Large Language Model Meta AI) further exemplifies this trend, with LLAMA being optimized for research and efficiency in large-scale natural language understanding tasks. While the evolution of Language Models has unlocked unprecedented capabilities in NLP, their effectiveness in real-world applications depends critically on how they are instructed, giving rise to the essential discipline of prompt engineering.

2.3 Prompt Engineering

Prompts serve as the input instructions provided by users to large language models (LLMs), guiding them toward producing desired outputs. It is important to recognize that LLMs may generate

varied responses based on the specific structure and wording of a prompt. Sometimes, the responses may be overly generic, vague, or irrelevant, a phenomenon referred to as "LLM hallucinations (Bender et al. [5]) and (Marcus [37]), highlighting how these models can generate misleading or inaccurate information due to over-reliance on probabilistic predictions rather than factual data.

To mitigate such issues, prompt engineering has emerged as a pivotal technique, focusing on the strategic development and optimization of task-specific instructions (prompts) to guide pre-trained LLMs toward generating high-quality, relevant responses Min et al. [42]. Prompt engineering enables users to control the model's outputs by fine-tuning the prompt's structure, which can significantly improve both the quality and utility of the results. The discipline of prompt engineering has been extensively studied and popularized in various works, including (Liu et al. [33]), (Tonmoy et al. [51]), and (Chen et al. [7]).

2.4 Related Work

Authors of (Marques et al. [39]) have studied the role of LLMs in SRE by analyzing various studies and integrating ChatGPT into the SRE process. It showed that the SRE process improved in brainstorming and creativity, providing real-time feedback, and fostering collaboration through diverse perspectives. It reduces human errors in documentation and enhances quality with accurate and unambiguous outputs. LLMs resulted in cost savings, higher productivity, and better project management overall, however, they face limitations, including potential biases from training data, the risk of hallucinations, and difficulties in explicability. Lack of contextual understanding necessitates human oversight to clarify requirements and prevent over-reliance on generated outputs. The authors discussed some future directions, including the exploration of new prompt construction techniques tailored for each stage of software requirement development, and the usage of external knowledge bases, or human-in-the-loop verification, can help ensure logical and factual accuracy in generated outputs.

According to a survey (Hemmat et al. [22], on the usage of LLMs in SRE, covering the limitations and challenges faced.

1. **Domain Understanding Limitations:** LLMs frequently exhibit deficiencies in domain-specific knowledge, resulting in misinterpretations of requirements. Key issues include failure to incorporate organizational policies and insufficient contextual awareness for specialized tasks.

2. **Output Reliability Deficits:** Studies document persistent quality concerns, such as vague or incomplete outputs and factual hallucinations, wherein models generate plausible but incorrect information, necessitating rigorous manual validation.

3. **Prompt Engineering Constraints:** Effective prompt design remains nontrivial due to token limitations and sensitivity to input phrasing. Domain-agnostic prompts often yield ill-formed requirements, underscoring the need for context-aware structuring.

4. **Methodological Limitations:** Experimental reproducibility is hampered by ad hoc hyperparameter selection and unoptimized setups, potentially compromising model adaptability and performance in RE contexts.

5. **Structural Inconsistencies:** LLMs frequently produce syntactically flawed outputs, including type mismatches in formal specifications and erroneous operator usage in code generation, demanding post-hoc correction.

Through an analysis of 28 studies, Green and Taylor [20] derived 36 prompt engineering guidelines for LLM use in SRE. The study found that LLMs are helpful for tasks like requirements verification and consistency checks, where template-based prompts enhance traceability and usability. However, significant limitations persist, particularly in requirements analysis and elicitation. LLMs struggle with ambiguous terminology (e.g., vague "context" definitions), circular contextual dependencies, and output instability—generating inconsistent or oversimplified results even with fixed inputs. Their validation capabilities are inherently limited, as they cannot objectively assess correctness and falter in late-stage technical assessments. While templates provide structure, they fail to address core challenges like restricted reasoning abilities, low feedback confidence, and reproducibility issues, which hinder complex analysis. Further, LLMs often misalign with stakeholder needs due to inadequate domain adaptation, superficial

reasoning patterns, and systemic mismatches between generated outputs and implementation realities. These constraints suggest that domain-specific fine-tuning or hybrid approaches (e.g., integrating general guidelines with domain-oriented prompts) may be necessary to improve LLMs' reliability in SRE, particularly for nuanced tasks like analysis and elicitation, where current performance remains inconsistent.

In the paper (Sahoo et al. [49]), the authors explore prompt engineering as a means of enhancing the capabilities of pre-trained large language models (LLMs). This approach focuses on strategically designing task-specific instructions, known as prompts, to guide model behavior without the need to update model parameters. The paper categorizes 29 distinct prompt engineering techniques according to their targeted functionalities, shedding light on the strengths and limitations of each technique. Despite significant successes, challenges such as biases, factual inaccuracies, and gaps in interpretability persist, highlighting the need for continued investigation and the development of effective mitigation strategies. Looking ahead, the authors pointed to some directions, addressing new tasks without additional training data, enhancing reasoning and logic, reducing hallucinations, optimizing user interaction, and ensuring consistency, coherence, and efficiency through self-reflection.

3 Language Model Challenges in SRE

We have identified different challenges for using LLMs in Software Requirements Engineering, some were related to the LLMs themselves, others were related to the prompts, and some were related to the nature of SRE tasks. It's not in the scope of this paper to discuss the internal structure or architecture of the LLM itself, nor the NLP or AI algorithms used within it. A total of 6 technical and 2 research limitations were identified, among others, as to why SRE practitioners are reluctant to adopt LLM in the field. Moving forward, we will use (TL) to refer to technical limitations and (RL) for research limitations.

3.1 Technical Issues

1. **TL1: Security & Privacy Risks**

   This is the most critical issue and threat mentioned, as using LLMs poses inherent data exposure risks through data leakage and unsecured API integrations, particularly when handling sensitive requirements. These vulnerabilities may violate compliance regimes and erode stakeholder trust in regulated domains.

2. **TL2: Unreliable Output Quality & Formatting**

   LLM models frequently generate incorrect statements or structurally flawed technical specifications or documentation. Such deficiencies necessitate rigorous manual validation, increasing the need for manual verification costs and risking defective system deployments.

3. **TL3: Context & Domain Understanding Gaps**

   LLMs lack mechanisms to internalize organizational policies or domain-specific constraints during requirements generation. This often produces non-compliant outputs requiring substantial post-hoc revision, delaying development cycles. This is one of the most painful points to any LLM usage since they are trained on a very large corpus.

4. **TL4: Computational & Operational Costs**

   The resources needed to create or educate LLMs can not be supported by the SRE practitioners. The resource intensity of fine-tuning and inference creates prohibitive scalability challenges for many teams. These economic barriers limit practical adoption despite the technology's theoretical benefits.

5. **TL5: Prompt Engineering Challenges**

   Model performance exhibits extreme sensitivity to minor prompt phrasing variations, demanding specialized expertise. This dependency introduces implementation delays and organizational reliance on scarce LLM-proficient personnel.

6. **TL6: Reasoning & Analysis Limitations**

   LLMs can not perform deductive reasoning or rigorous analysis comparable to formal methods. Consequently, their utility remains restricted to supplementary tasks rather than critical decision-making processes. This is due to the lack of specific training given to the LLM since it needs to be of a general use case.

3.2 Research Issues

1. **RL1: Dataset availability**

   Our literature review reveals that existing studies in this domain lack experimentation with dedicated datasets for requirement engineering tasks. However, through examination of open-source repositories, we identified specialized datasets (OpenScience Community [45], Dalpiaz et al. [11]) that have been exclusively utilized for requirement elicitation using NLP techniques. This presents both an opportunity to validate prior work and a limitation in current research methodologies.

2. **RL2: Evaluation Methods**

   The assessment of LLM applications in Software Requirements Engineering faces significant methodological challenges due to three interrelated constraints: the absence of standardized benchmark datasets with expert-validated ground truth annotations for most SRE tasks, the lack of established quantitative metrics to objectively measure output quality beyond subjective expert judgment, and an over-reliance on limited-scale human evaluations that incur substantial costs while potentially introducing individual biases and failing to represent the full spectrum of SRE scenarios. These limitations collectively undermine the reproducibility, scalability, and objective validation of research findings in this domain.

These challenges open the way for the following research questions:

1. ***RQ1:*** *How can language models (LMs) overcome computational, domain, and reliability limitations in Software Requirements Engineering (SRE)?*

2. ***RQ2:*** *How can modular frameworks enhance the security and scalability of LM-augmented SRE workflows?*

3. ***RQ3:*** *What prompting strategies ensure accurate, context-aware requirements generation and analysis?*

Our analysis reveals a clear dichotomy in LLM challenges: constraints and restrictions (TL1, TL4) versus inherent model capabilities (TL2, TL6). Furthermore, we identify two critical dimensions of human-LLM interaction – effective communication through prompt engineering (TL5) and domain knowledge limitations (TL3) – that collectively shape the practical utility of these systems. These findings are further contextualized by two unresolved research issues: the absence of dedicated datasets for requirement engineering tasks (RL1) and fundamental limitations in current evaluation methodologies (RL2). By analyzing established research in language modeling and prompt-based interaction paradigms, we propose a conceptual framework for potential LM applications in Software Requirements Engineering. This theoretical investigation establishes foundational insights to guide future empirical validation in SRE contexts. Systematic incorporation of existing datasets with preliminary ground truth annotations and established NLP evaluation metrics, particularly for requirement elicitation tasks. These datasets will be extended and adapted to ensure comprehensive coverage of SRE scenarios. Implementation of multimodal validation strategies, beginning with expert assessments of framework-generated outputs. Verified results will be archived as refined ground truth datasets, creating a cyclical process that enhances both current validation rigor and future research reproducibility.

4 Conceptual Framework for Language Models in SRE

Building on the identified challenges of applying Language Models (LMs) to Software Requirements Engineering (SRE) (Section 3), this section formalizes a conceptual framework to address these limitations through structured theoretical integration. By synthesizing foundational LM architectures (Section 2.2), prompt engineering paradigms (Section 2.3), and SRE-specific task requirements, we propose a 4 parts model that: (1) maps LM constraints to SRE problem categories (TL1, TL4), (2) systematic strategies to address LLM hallucinations and capability gaps (TL2, TL6), (3) formulate prompts (TL5) to reduce hallucinations and reach more desired output, and (4) incorporates domain-knowledge adaptation mechanisms (TL3). The framework explicitly avoids empirical validation, instead providing a scaffold for future applied research.

4.1 Addressing LM constrains

To address Language Models constraints for LLM mentioned in (TL1 and TL4), which are related to the security concerns and cost of training and operations. We searched for approaches that keep the LM locally controlled to reduce the risk of data exposure to external parties, as well as a model that can be easily trained and operated without consuming vast resources or cost. This highlights the need to shift focus toward developing smaller, yet powerful, language models that are more efficient and feasible to deploy (Hu et al. [27]). Small Language Models (SLMs) offer a lightweight yet capable alternative to large language models (LLMs), balancing efficiency and accessibility with typically under 7 billion parameters, enabling deployment on personal devices without GPUs like tinyLlamma (Zhang et al. [61]). Unlike LLMs, which rely on massive scale, SLMs democratize NLP by reducing costs, lowering resource demands, and allowing faster experimentation for specialized applications. Their practicality makes them ideal for everyday use as well as locally deployed, while maintaining strong language understanding.

SLMs achieve strong performance by training smaller models on more tokens than traditional scaling laws suggest (Hoffmann et al. [25]), emphasizing optimized data utilization over sheer model size, as demonstrated in works like (Touvron et al. [53]). Researchers have also explored fine-tuning or distilling LLMs into task-specific Small Language Models (SLMs) (Fu et al. [19], Ho et al. [24], Hsieh et al. [26]). By focusing on inference constraints and efficient data allocation, SLMs bridge the gap between compact design and robust functionality, enabling their integration into resource-constrained environments while retaining competitive NLP capabilities. Despite their efficiency, SLMs still face two critical gaps: (1) weaker complex reasoning abilities compared to LLMs, and (2) limited capacity for knowledge-intensive tasks due to their smaller parameter size. Addressing these gaps requires innovations in both model architecture and training methodologies to enhance performance without sacrificing efficiency (Kang et al. [29]).

4.2 Incorporating Domain Knowledge to LM

LLMs are trained over a large language corpus with a lot of human knowledge, reducing the focus and increasing the possibilities of hallucinations. Knowledge-Augmented Language Models (LMs) boost Small Language Models (SLMs) by dynamically retrieving relevant information from external knowledge bases (e.g., Wikipedia), enabling factually grounded responses without requiring memorization. Approaches like Knowledge-Augmented Reasoning Distillation (KARD) (Kang et al. [29]) further enhance SLMs by fine-tuning them with LLM-generated rationales and task-specific external knowledge, combining parametric reasoning skills with non-parametric memory, allowing efficient, accurate performance in knowledge-intensive tasks despite smaller parameter counts.

4.3 Extending LM cabability

While language models excel at processing natural language inputs, their ability to generate structured outputs or manage complex, multi-step tasks remains limited without explicit guidance. This necessitates a systematic approach to control output formatting and orchestrate intricate workflows effectively.

LangChain is a modular framework designed to streamline the development of scalable, context-aware applications powered by language models (LMs). By seamlessly integrating external data sources, retrieval-augmented generation (RAG), and secure API interactions, it bridges the gap between LM capabilities and real-world deployment, addressing critical challenges like state management, contextual understanding, and security. The framework provides comprehensive tools for diverse use cases, including autonomous agents, chatbots, data extraction, and structured data analysis, empowering developers across various domains to build adaptable and secure LLM-driven solutions with efficiency (Topsakal and Akinci [52], Mavroudis [40], Duan [16]). Despite its advantages, LangChain's reliance on external integrations introduces critical security considerations, particularly data exposure and dependency risks, which demand rigorous safeguards in sensitive domains like healthcare or finance. While modularity enables flexibility, it also amplifies system complexity, necessitating robust security protocols to ensure data integrity and privacy without compromising functionality (Topsakal and Akinci [52]).

4.4 Prompts formulation

To enable users to leverage language models (LMs) effectively for Software Requirements Engineering (SRE) tasks, we systematically investigated and developed structured prompt engineering techniques

to optimize LM interactions and outputs.

4.4.1 Prompt Guidelines

Building on the work of (Green and Taylor [20]), which outlined 36 prompt engineering guidelines to use in SRE, we identified a condensed set of four primary guideline categories to optimize prompt design in software requirements engineering. These guidelines need to be followed in any usage of LLM for SRE tasks.

1. **Context**: Providing relevant context in the prompt is essential for enhancing result quality and reducing instances of hallucinations.

2. **Language**: Using clear, concise, and grammatically correct English, along with short, focused sentences, improves the LLM's comprehension and response accuracy.

3. **Examples**: Including examples in prompts aids in guiding the LLM, particularly when tasks are ambiguous, and strengthens the effectiveness of zero-shot prompts.

4. **Keywords**: Some keywords can enhance the LLM's ability to process complex queries and maintain logical coherence.

These four categories encompass the broader guideline defined in (Green and Taylor [20]). Context and language are fundamental to any prompt strategy, they can change the scope of the result and guide to different outputs. Using examples can help in fine-tuning the LLM by teaching it how to handle the task, This was elaborated more in (Brown et al. [6]) paper, which discussed the few-shot prompt and how the example can enhance the prompt's result. Keywords like "think step by step" and others can greatly impact how the LLM will work out the result.

4.4.2 Prompt Strategies

Improving Large Language Model (LLM) prompt performance can be broadly categorized into two approaches: *(1) human-side prompt engineering*, which focuses on optimizing the input prompts provided by users, and *(2) model-side architectural enhancements*, which modify the LLM's internal mechanisms, Figure- 1. In this work, we focus on the former, specifically, how to enhance prompts from the human (sender) side to maximize LLM effectiveness. Prompting strategies can be further divided into manual and automatic approaches:

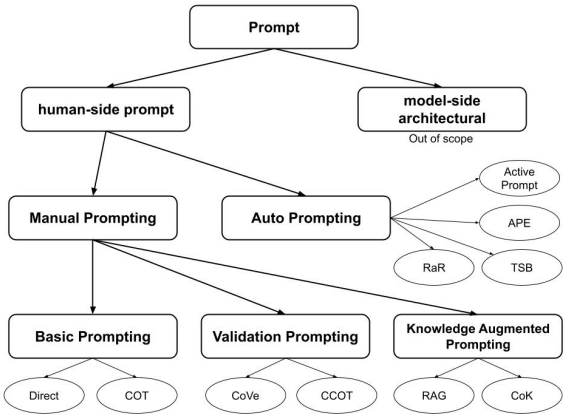

Figure 1: Prompt Strategies categories

Manual prompts are crafted directly by humans, often through iterative testing (e.g., zero-shot or chain-of-thought prompting). Automatic prompts leverage LLMs themselves to generate or refine inputs. This includes methods like: Active Prompting (Diao et al. [15]), Automatic Prompt Engineer (APE) (Zhou et al. [64]), Take Step Back (TSB) (Zheng et al. [63]), or Rephrase and Respond (RaR) (Deng et al. [12]), where LLMs suggest improvements to manually drafted prompts.

Another dimension of prompting involves integrating external knowledge sources. For instance, Retrieval-Augmented Generation (RAG) (Lewis et al. [31]), Chain of knowledge (CoK) (Li et al. [32]) dynamically pulls information from external databases to ground responses in factual data. They use external repositories not only as sources but also for real-time validation of LLM outputs. To mitigate errors, recent work has introduced validation-focused prompting strategies: Chain of Verification (CoVe) (Dhuliawala et al. [14]) and Contrastive Chain-of-Thought (CCOT) (Chia et al. [8]), that embed self-checking mechanisms within prompts, forcing the LLM to validate its output. Traditional methods like direct (zero-shot) prompting (Radford et al. [47]) or Chain-of-Thought (CoT) (Wei et al. [57]) remain foundational. CoT, for example, explicitly structures the LLM's reasoning process into step-by-step sequences, significantly improving performance on complex tasks. However, the field is rapidly evolving toward hybrid approaches that combine manual craftsmanship, automated optimization, and external knowledge integration to address the limitations of any single method.

4.5 Takeaways

Based on the above findings, we can summarize the difference between the different approaches of LM as well as the prompt design as follows.

Large Language Models (LLMs) excel in complex reasoning and versatility but are costly and environmentally intensive, making them impractical for many applications. Small Language Models (SLMs) address these issues with efficient, lightweight designs suitable for edge deployment, though they lag in reasoning and knowledge retention. Knowledge-Augmented Language Models (KALMs) bridge this gap by integrating external knowledge bases, enhancing domain-specific accuracy without sacrificing efficiency. LangChain, as a framework, complements all three by enabling modular, context-aware applications through tools such as RAG, memory, and agents, although it introduces added complexity and security considerations. Together, these technologies form a spectrum of solutions balancing performance, cost, and deployability, with SLMs and KALMs democratizing access to advanced NLP and LangChain streamlining real-world integration.

Prompt construction for language models follows three primary approaches: (1) manual user input, (2) retrieval from template repositories, or (3) automatic generation using auto-prompting strategies (e.g., Active Prompting, APE, TSB, RaR). For specialized applications like SLMs or Knowledge-Augmented LMs, techniques such as RAG and Chain-of-Knowledge (CoK) prove essential by enabling dynamic data retrieval and integration. Foundational prompting methods like zero-shot and Chain-of-Thought can be augmented through example-based refinement, while verification frameworks like CoVe and CCOT provide critical output validation across all prompting strategies, serving as universal safeguards for LM reliability.

5 Limitations and future directions

The current study only proposes a hypothetical framework without a practical implementation to prove the concept. In our research above, we studied only existing research, overlooking existing commercial tools that may exist to support the SRE process. Future work should focus on the following:

1. Expanding the KALM knowledge base to cover additional SRE subdomains.
2. Developing standardized prompt templates for industry-specific use cases.
3. Optimizing the auto-prompting pipeline for complex, multi-stage SRE workflows.
4. Compare our proposed framework to any existing commercial tools.

6 Conclusion

This paper provides an insight into current research on Language Models (LMs) in the Software Requirements Engineering (SRE) domain. Key challenges, such as security, cost, relevance, control, and domain knowledge, hamper the effective usage of Large Language Models (LLMs) in SRE. Additionally, limitations related to datasets and evaluation metrics present obstacles for researchers, often necessitating reliance on expert judgment rather than established ground truths.

To address these challenges and limitations, we propose a conceptual framework to mitigate these issues while serving as a reference for future research. The framework integrates multiple specialized Knowledge-Augmented Language Models (KALMs) with Small Language Models (SLMs) within a LangChain ecosystem, offering a comprehensive solution that will mitigate security risks, optimize operational costs, enhance contextual relevance, and ensure output control.

By implementing knowledge-augmented prompting techniques, such as Retrieval-Augmented Generation (RAG) and Chain-of-Knowledge, alongside KALMs, and by maintaining a repository of fine-tuned, auto-generated prompt templates for common SRE tasks, the framework significantly improves system reliability. Furthermore, incorporating validation strategies (e.g., Chain-of-Verification, CCOT) as a mandatory output-checking layer ensures robust and verifiable results.

This approach establishes a foundation for trustworthy, efficient, and scalable AI-assisted SRE practices while overcoming the limitations of current LLM applications. Beyond serving as an analytical tool, the proposed framework also facilitates the generation of standardized evaluation resources, contributing to methodological consistency in future research.

References

[1] W. Alhoshan, A. Ferrari, and L. Zhao. 2023. Zero-shot learning for requirements classification: An exploratory study. *Information and Software Technology*, 159:107202.

[2] S. Arora, A. Narayan, M. F. Chen, and et al. 2023. Ask me anything: A simple strategy for prompting language models. In *The Eleventh International Conference on Learning Representations*.

[3] Y. Bang, S. Cahyawijaya, N. Lee, and et al. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023*.

[4] L. Bencheikh and N. Höglund. 2023. *Exploring the Efficacy of ChatGPT in Generating Requirements: An Experimental Study*. Bachelor's thesis, Chalmers University of Technology, Göteborg, Sweden.

[5] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 610–623.

[6] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901. NeurIPS 2020.

[7] Q. Chen, H. Zou, and S. Yang. 2023. Advancements in prompt engineering for large language models: A survey of techniques and applications. *Journal of Machine Learning Research*, 24(101):1–20.

[8] Yew Ken Chia, Guizhen Chen, Luu Anh Tuan, Soujanya Poria, and Lidong Bing. 2023. Contrastive chain-of-thought prompting. *arXiv preprint arXiv:2311.09277*. ArXiv:2311.09277 [cs.CL].

[9] F. Dalpiaz, A. Ferrari, X. Franch, and C. Palomares. 2018. Natural language processing for requirements engineering: The best is yet to come. *IEEE Software*, 35(5):115–119.

[10] F. Dalpiaz and N. Niu. 2020. Requirements engineering in the days of artificial intelligence. *IEEE Software*, 37:7–10.

[11] Fabiano Dalpiaz, Davide Dell'Anna, Fatma Başak Aydemir, and Sercan Çevikol. 2019. Supplementary material for "requirements classification with interpretable machine learning and dependency parsing".

[12] Yihe Deng, Weitong Zhang, Zixiang Chen, and Quanquan Gu. 2023. Rephrase and respond: Let large language models ask better questions for themselves. *arXiv preprint arXiv:2311.04205*.

[13] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv*, 1810.04805.

[14] Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2023. Chain-of-verification reduces hallucination in large language models. *Preprint*, arXiv:2309.11495. ArXiv:2309.11495 [cs.CL].

[15] Shizhe Diao, Pengcheng Wang, Yong Lin, and Tong Zhang. 2023. Active prompting with chain-of-thought for large language models. *arXiv preprint arXiv:2302.12246*.

[16] Zhihua Duan. 2023. Application development exploration and practice based on langchain+chatglm+rasa. In *2023 2nd International Conference on Cloud Computing, Big Data Application and Software Engineering (CBASE)*, pages 282–285.

[17] A. Fan, B. Gokkaya, M. Harman, M. Lyubarskiy, S. Sengupta, S. Yoo, and J.M. Zhang. 2023. Large language models for software engineering: Survey and open problems. *arXiv preprint arXiv:2310.03533*.

[18] A. Ferrari, A. Esuli, and S. Gnesi. 2018. Identification of cross-domain ambiguity with language models. In *2018 5th International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)*, pages 31–38. IEEE.

[19] Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023. Specializing smaller language models towards multi-step reasoning. *arXiv preprint arXiv:2301.12726*.

[20] Alice Green and Bob Taylor. 2023. Prompt engineering guidelines for llms in requirements engineering. In *Proceedings of the 2023 IEEE International Conference on Requirements Engineering*, pages 45–50.

[21] Y. Hao, Z. Chi, L. Dong, and F. Wei. 2022. Optimizing prompts for text-to-image generation. *arXiv preprint arXiv:2212.09611*.

[22] A. Hemmat, M. Sharbaf, K. Lano, and S. Y. Tehrani. 2025. Research directions for using LLM in software requirement engineering: A systematic review. *Frontiers in Computer Science*, 7:1519437.

[23] T. Hey, J. Keim, A. Koziolek, and W. F. Tichy. 2020. Norbert: Transfer learning for requirements classification. In *2020 IEEE 28th International Requirements Engineering Conference (RE)*, pages 169–179. IEEE.

[24] Namgyu Ho, Laura Schmid, and Se-Young Yun. 2023. Large language models are reasoning teachers. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14852–14882, Toronto, Canada. Association for Computational Linguistics.

[25] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. 2022. Training compute-optimal large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 30016–30030.

[26] Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8003–8017, Toronto, Canada. Association for Computational Linguistics.

[27] Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, Xinrong Zhang, Zheng Leng Thai, Kaihuo Zhang, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. Minicpm: Unveiling the potential of small language models with scalable training strategies. *Preprint*, arXiv:2404.06395.

[28] International Institute of Business Analysis (IIBA). 2015. *A Guide to the Business Analysis Body of Knowledge (BABOK® Guide)*, 3rd edition. International Institute of Business Analysis, Toronto, ON, Canada.

[29] Minki Kang, Seanie Lee, Jinheon Baek, Kenji Kawaguchi, and Sung Ju Hwang. 2023. Knowledge-augmented reasoning distillation for small language models in knowledge-intensive tasks. *Preprint*, arXiv:2305.18395. ArXiv:2305.18395 [cs.CL].

[30] K. Kaur, P. Singh, and P. Kaur. 2020. A review of artificial intelligence techniques for requirement engineering. In *Computational Methods and Data Engineering: Proceedings of ICMDE 2020, Volume 2*, pages 259–278.

[31] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and Sebastian Riedel. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

[32] Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Shafiq Joty, Soujanya Poria, and Lidong Bing. 2023. Chain-of-knowledge: Grounding large language models via dynamic knowledge adapting over heterogeneous sources. *Preprint*, arXiv:2305.12769.

[33] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in nlp. *ACM Computing Surveys (CSUR)*, 55(9):1–35.

[34] V. Liu and L. B. Chilton. 2022. Design guidelines for prompt engineering text-to-image generative models. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–23.

[35] X. Luo, Y. Xue, Z. Xing, and J. Sun. 2022. Prcbert: Prompt learning for requirement classification using bert-based pretrained language models. In *37th IEEE/ACM International Conference on Automated Software Engineering*, pages 1–13.

[36] P. Maddigan and T. Susnjak. 2023. Chat2vis: Generating data visualizations via natural language using chatgpt, codex, and gpt-3 large language models. *arXiv preprint arXiv:2302.02094*.

[37] G. Marcus. 2022. Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631v2*.

[38] N. Marques, R. R. Silva, and J. Bernardino. 2024. Using chatgpt in software requirements engineering: A comprehensive review. *Future Internet*, 16(6):180.

[39] N. Marques, R. R. Silva, and J. Bernardino. 2024. Using chatgpt in software requirements engineering: A comprehensive review. *Future Internet*, 16(6):180.

[40] Vasilios Mavroudis. 2024. LangChain v0.3. Working paper or preprint.

[41] T. Mikolov, K. Chen, G. Corrado, and J. Dean. 2013. Efficient estimation of word representations in vector space. *arXiv*, 1301.3781.

[42] B. Min, H. Ross, E. Sulem, A.P.B. Veyseh, T.H. Nguyen, O. Sainz, E. Agirre, I. Heintz, and D. Roth. 2023. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*. Just Accepted.

[43] G. Mogyorodi. Requirements-based testing: An overview. In *Proceedings of the TOOLS Conference*, pages 286–295.

[44] OpenAI. 2023. Openai prompt design guidelines. `https://platform.openai.com/docs/guides/completion/promptdesign`. Accessed: 2023-03-10.

[45] OpenScience Community. 2023. Openscience requirements engineering dataset. `https://openscience.us/repo/requirements/`.

[46] Project Management Institute (PMI). 2015. *Business Analysis for Practitioners: A Practice Guide*. Project Management Institute, Newtown Square, PA, USA.

[47] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

[48] K. Ronanki, C. Berger, and J. Horkoff. 2023. Investigating chatgpt's potential to assist in requirements elicitation processes. In *Proceedings of the 49th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*.

[49] Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. 2024. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2406.06608*.

[50] A. Sainani, P. R. Anish, V. Joshi, and S. Ghaisas. 2020. Extracting and classifying requirements from software engineering contracts. In *2020 IEEE 28th International Requirements Engineering Conference (RE)*, pages 147–157. IEEE.

[51] M. Tonmoy, R. Jahan, and T. Ahmed. 2024. Strategic design of task-specific prompts for large language models. *Journal of Artificial Intelligence Research*, 65:158–172.

[52] Oguzhan Topsakal and T. Cetin Akinci. 2023. Creating large language model applications utilizing langchain: A primer on developing llm apps fast. *International Conference on Applied Engineering and Natural Sciences*, 1:1050–1056.

[53] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

[54] Muhammad Aminu Umar and Kevin Lano. 2024. Advances in automated support for requirements engineering: A systematic literature review. *Requirements Engineering*, 29(2):177–207.

[55] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, and I. Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*, pages 6000–6010.

[56] Bingyang Wei. 2023. Requirements are all you need: From requirements to code with llms. In *Proceedings of the 2023 IEEE International Conference on Software Engineering*, Fort Worth, USA. Department of Computer Science, Texas Christian University, IEEE.

[57] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. NeurIPS 2022.

[58] L. Westfall. 2005. Software requirements engineering: What, why, who, when, and how. *Software Quality Professional*, 7:17–23.

[59] Karl Wiegers and Joy Beatty. 2013. *Software Requirements*, 3rd edition. Microsoft Press, Redmond, WA, USA.

[60] J. Winkler and A. Vogelsang. 2016. Automatic classification of requirements based on convolutional neural networks. In *2016 IEEE 24th International Requirements Engineering Conference Workshops (REW)*, pages 39–45. IEEE.

[61] Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. Tinyllama: An open-source small language model.

[62] Hao Zhen, Yucheng Shi, Yongcan Huang, Jidong J. Yang, and Ninghao Liu. 2024. Leveraging large language models with chain-of-thought and prompt engineering for traffic crash severity analysis and inference. *arXiv*, 2408.04652.

[63] Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2023. Take a step back: Evoking reasoning via abstraction in large language models. *arXiv preprint arXiv:2310.06117*.

[64] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*.