BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

Leveraging LLMs for Coverage Analysis from High to Low-Level Requirements

Sian Ashsad
200042151

Department of Computer Science and Engineering
Islamic University of Technology
September, 2025

Leveraging LLMs for Coverage Analysis from High to Low-Level Requirements

Sian Ashsad
200042151

Department of Computer Science and Engineering
Islamic University of Technology
September, 2025

Declaration of Candidate

This is to certify that the work presented in this thesis is the outcome of the analysis and experiments carried out by **Sian Ashsad** under the supervision of **Shohel Ahmed**, Assistant Professor, Department of Computer Science and Engineering, Islamic University of Technology, Dhaka, Bangladesh. It is also declared that neither this thesis nor any part of it has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others have been acknowledged in the text and a list of references is given.

Shohel Ahmed
Assistant Professor
Department of Computer Science and Engineering
Islamic University of Technology (IUT)
Date: September 30, 2025

Sian Ashsad
Student ID: 200042151
Date: September 30, 2025

Contents

List of Abbreviations

HLR	High Level Requirments
LLR	Low Level Requirements
LLM	Large Language Model
RSA	Requirement Satisfaction Assessment
CoT	Chain of Thought
zS	Zero Shot
TP	True Positive
FP	False Positive
FN	False Negative

Acknowledgement

I would like to express my sincere gratitude to my supervisor, Mr. Shohel Ahmed, for his invaluable guidance, support, and encouragement throughout the course of this thesis. His insightful feedback and continuous mentorship have been crucial to the completion of this work.

I would also like to extend my heartfelt thanks to the faculty members who provided their expertise, constructive suggestions, and constant support during this journey. Their contributions have been instrumental in shaping and enriching my research experience.

Finally, my deepest appreciation to the Department of Computer Science and Engineering for providing me with the resources, opportunities, and a stimulating academic environment that made this work possible. Their continuous support and dedication to academic excellence have been fundamental to my learning and growth.

Abstract

The increasing capabilities of Large Language Models (LLMs) have opened new avenues for automating critical tasks in software and systems engineering, particularly in the generation and analysis of requirements. This thesis investigates the potential of LLMs to generate Low-Level Requirements (LLRs) from existing High-Level Requirements (HLRs), aiming to evaluate how effectively these models can replicate human-derived requirements. The study employs multiple state-of-the-art LLMs to generate LLRs using various prompting strategies, followed by a systematic evaluation of coverage and traceability through expert validation and alignment with established academic criteria.

The findings of this research provide insights into the strengths and limitations of LLMs in capturing detailed, actionable system specifications. While LLMs demonstrate significant potential for automating certain aspects of requirements engineering, human expertise remains essential for ensuring completeness, accuracy, and contextual relevance. By highlighting areas where automation can enhance efficiency and where human oversight is necessary, this work contributes to a deeper understanding of LLM capabilities in requirements engineering and offers practical recommendations for integrating these models into professional practice to improve requirement traceability, consistency, and quality.

Chapter 1

Introduction

Requirements engineering, or RE, is a crucial component of software engineering since it forms the basis for all other development phases. Traditional requirements documentation uses natural language [18] and can range from casual explanations to formalized formats like use cases.

Researchers are motivated to use natural language processing technology for a variety of requirements engineering tasks due to the textual character of requirements [28]. The differing levels of abstraction between high-level and low-level requirements, their reliance on natural language, and the frequently disparate terminology employed present a problem for supporting automatic reviews for requirements coverage. This difficulty can now be addressed in novel ways thanks to the emergence of Large Language Models (LLMs), which have been trained on large text corpora and are able to contextualize both high-level and low-level requirements [23].

In recent years, the application of Large Language Models (LLMs) in software engineering tasks has gained significant attention. This thesis explores the use of LLMs to automate and analyze requirement coverage from High-Level Requirements (HLR) to Low-Level Requirements (LLR). By leveraging state-of-the-art models namely Gemini 2.0 Flash, Meta Llama 3 8B and DeepSeek 7B chat, this work aims to generate LLRs from HLRs and assess the coverage achieved in comparison to existing human-curated mappings.

1.1 Background

The accurate translation of system requirements from a high-level form to detailed, actionable specifications is a critical task in the development of complex software sys-

tems. High-Level Requirements (HLRs) describe what a system must accomplish, often in broad or business-oriented terms, while Low-Level Requirements (LLRs) break down these goals into precise, technical directives that guide implementation. Ensuring requirement coverage—that is, verifying that all HLRs are adequately addressed by corresponding LLRs—is essential for building reliable, compliant, and high-quality systems.

Traditionally, the process of mapping HLRs to LLRs has relied heavily on human expertise, which is time-consuming, labor-intensive, and prone to oversight [26]. However, recent advances in Artificial Intelligence (AI), particularly in Large Language Models (LLMs) such as Gemini 2.0 Flash [17], Meta Llama 3 8B Instruct [1] and DeepSeek 7B Chat [7], offer new opportunities to automate and support this critical activity. LLMs, trained on vast amounts of text data, have shown remarkable abilities in understanding and generating human-like text, making them promising tools for requirement engineering tasks.

This research investigates the use of LLMs to generate LLRs from HLRs and evaluates the extent to which these generated LLRs provide complete and correct coverage compared to human-created mappings. The study uses real-world datasets from the Center of Excellence for Software & Systems Traceability (CoEST) [6], including projects from domains like healthcare (e.g., HIPAA requirements) [3] and aerospace (e.g., NASA's CM1 dataset) [14], where requirement accuracy is critical for safety, compliance, and system integrity. By exploring various prompting techniques—including Zero-Shot prompting, Zero-Shot with Explanation, and Chain of Thought reasoning—this thesis also replicates and extends existing research, aiming to deepen our understanding of how prompting strategies influence LLM performance in requirement coverage tasks.

The ability for automation of certain steps of the requirement engineering process could greatly improve system development's efficiency and dependability as software systems continue to grow in complexity and societal significance, especially in industries like cybersecurity, aerospace, and healthcare. Through this study, we aim to contribute to the larger endeavor of securely and effectively incorporating AI into crucial phases of software engineering.

1.2 Motivations and Scope

As software systems become increasingly complex and mission-critical—especially in sectors like healthcare, aerospace, and cybersecurity—the need for precise and com-

plete requirements engineering has never been greater. Any gaps or errors in the mapping from High-Level Requirements (HLRs) to Low-Level Requirements (LLRs) can lead to costly failures, safety risks, and regulatory non-compliance. Traditionally, ensuring complete requirement coverage has relied heavily on manual processes, which are time-consuming, expensive, and prone to human error.

At the same time, recent advancements in Artificial Intelligence, particularly in the capabilities of Large Language Models (LLMs), present an opportunity to rethink and automate parts of this critical engineering task. LLMs have shown exceptional potential in tasks involving comprehension, reasoning, and text generation—skills closely aligned with requirement interpretation and breakdown. Leveraging these models could not only reduce manual effort but also improve the consistency and reliability of requirements traceability.

This thesis is motivated by the possibility of harnessing state-of-the-art LLMs to assist in requirement engineering, particularly in bridging the gap between HLRs and LLRs. By evaluating the coverage quality of LLM-generated LLRs against trusted human-curated datasets, and by exploring the effects of different prompting strategies, this research aims to assess the practical readiness of LLMs for real-world requirement engineering tasks.

This thesis focuses on the use of Large Language Models (LLMs) for automating and analyzing the coverage of Low-Level Requirements (LLRs) generated from High-Level Requirements (HLRs). The work involves:

- Generating LLRs from HLRs using selected LLMs (Gemini 2.0 Flash, Meta Llama 3 8B and DeepSeek 7B Chat).
- Utilizing datasets from the Center of Excellence for Software & Systems Traceability (COeST), specifically CCHIT, CM1-NASA, GANTT, HIPAA, WARC, and IP.
- Applying and comparing three prompting strategies—Zero-Shot (zS), Zero-Shot with Explanation (zS+E), and Chain of Thought (CoT) [27] —to evaluate their impact on LLM performance.
- Measuring the quality of requirement coverage based on expert validation and established coverage criteria drawn from the literature.
- Replicating prior research methods to ensure comparability, while also extending the analysis with additional models and perspectives.

The thesis does not aim to create new datasets, propose new LLM architectures, or

perform fine-tuning of the models; rather, it focuses on evaluating existing general-purpose models and prompting techniques for their suitability in requirement engineering contexts.

1.3 Problem Statement

The manual derivation and validation of Low-Level Requirements (LLRs) from High-Level Requirements (HLRs) are critical yet labor-intensive tasks in software development, especially in domains requiring high assurance, such as healthcare and aerospace. Human-driven requirement coverage processes are time-consuming, susceptible to inconsistencies, and difficult to scale for large systems. Although Large Language Models (LLMs) have demonstrated strong capabilities in text understanding and generation, their effectiveness in generating accurate and complete LLRs from HLRs remains largely unexplored and invalidated in a systematic, traceability-focused context. There is a pressing need to assess whether LLMs, with appropriate prompting strategies, can reliably assist in bridging HLRs to LLRs while maintaining acceptable coverage quality, thus supporting the automation and enhancement of requirement engineering practices [23].

1.3.1 Research Objectives

- To evaluate the ability of selected Large Language Models (Gemini 2.0 Flash [17], Meta Llama 3 8B Instruct [1] and DeepSeek 7B Chat [7]) to generate Low-Level Requirements (LLRs) from High-Level Requirements (HLRs).
- To assess the requirement coverage provided by LLM-generated LLRs against established human-curated datasets using expert validation and literature-based criteria.
- To compare the effectiveness of different prompting strategies—Zero-Shot (zS), Zero-Shot with Explanation (zS+E), and Chain of Thought (CoT) [27] —in improving the quality of LLR generation.
- To replicate and extend existing research methodologies in the domain of requirements traceability, contributing insights into the readiness of LLMs for real-world requirement engineering tasks.

1.3.2 Research Questions

RQ1: How well can LLMs generate LLRs from HLRs using various prompting strategies?
This question investigates the capability of large language models (LLMs) to generate low-level requirements (LLRs) from high-level requirements (HLRs) using different prompting strategies, aiming to understand the potential and limitations of automated LLR generation.

RQ2: What is the coverage of LLM-generated LLRs when compared to human-generated LLRs across multiple datasets?
This question examines the coverage of LLM-generated LLRs in comparison to human-generated LLRs across multiple datasets, assessing how well the models capture the full set of requirements specified by humans.

RQ3: How does the use of different prompting strategies impact the quality of LLR generation?
This question explores how varying prompting strategies influence the quality of LLRs generated by LLMs, focusing on the role of prompt design in producing accurate, relevant, and comprehensive requirements.

1.4 Research Challenges

Automating the generation and validation of Low-Level Requirements (LLRs) from High-Level Requirements (HLRs) using Large Language Models (LLMs) introduces several nuanced challenges unique to the domain of requirement engineering and AI application in critical systems:

1. **Preserving Semantic Fidelity Between HLRs and LLRs:**
   One of the core challenges is ensuring that the LLM-generated LLRs accurately capture the intent, constraints, and conditions specified in the original HLRs. Small deviations in interpretation can lead to incomplete or incorrect system behavior, particularly in safety-critical domains like healthcare and aerospace.

2. **Evaluating Requirement Coverage Objectively:**
   Assessing the quality and completeness of generated LLRs is not straightforward. Unlike typical text generation tasks where fluency and coherence are sufficient, requirement coverage demands strict traceability, logical consistency, and domain-specific accuracy, which are difficult to measure automatically and often require expert evaluation.

3. **Handling Domain-Specific Language and Complexity:**
   HLRs and LLRs often contain specialized terminology, implicit assumptions, and layered technical details that general-purpose LLMs may not fully grasp. Successfully navigating domain-specific complexity without domain fine-tuning remains a major hurdle.

4. **Prompting Strategy Sensitivity:**
   The output quality of LLMs is highly sensitive to the prompting strategy employed. Crafting prompts that elicit structured, complete, and accurate LLRs consistently across diverse datasets is challenging, and small variations in prompt phrasing can significantly impact results.

5. **Dataset Variability and Generalization:**
   The datasets used (e.g., HIPAA, CM1-NASA, GANTT) span different domains and project types, each with distinct styles of requirements. Ensuring that LLM performance generalizes across these varied datasets, rather than overfitting to specific formats or structures, is a critical research obstacle.

6. **Absence of Standardized Benchmarks for Requirements Traceability via LLMs:**
   Unlike tasks such as summarization or translation, requirement engineering lacks well-established, standardized benchmarks for evaluating AI performance. This makes comparative analysis more complex and requires careful design of evaluation metrics and protocols.

By addressing these challenges, this research not only explores the capabilities of current LLMs but also highlights critical gaps and limitations that future AI models and methodologies must overcome to be effective in automated requirement engineering.

1.5 Contribution

Our work makes several novel contributions to the intersection of requirement engineering and artificial intelligence, specifically in the application of Large Language Models (LLMs) for requirement coverage analysis:

- **Evaluation of LLMs for Requirement Traceability:**
  A systematic evaluation of three state-of-the-art LLMs—Gemini 2.0 Flash, Meta Llama 3 8B and DeepSeek 7B Chat—on the task of generating Low-Level Requirements (LLRs) from High-Level Requirements (HLRs), using real-world datasets from the Center of Excellence for Software & Systems Traceability (CO-

eST) [6].

- **Integration and Analysis of Prompting Strategies:**
  Replication and extension of prior research by implementing and comparing three prompting strategies: Zero-Shot (zS), Zero-Shot with Explanation (zS+expl.), and Chain of Thought (CoT), highlighting their impact on the quality of generated requirements across different datasets.
- **Expert-Validated Requirement Coverage Assessment:**
  Coverage analysis of the LLM-generated LLRs through expert validation and established literature-based criteria, providing a grounded and rigorous evaluation rather than relying solely on automated or heuristic-based metrics.
- **Identification of LLM Strengths and Limitations in Requirements Engineering:**
  A detailed investigation into the capabilities and boundaries of general-purpose LLMs when applied to the structured and domain-specific task of requirement breakdown, offering valuable insights for both researchers and practitioners.
- **Creation of a Comparative Framework for Future Research:**
  The development of a comparative analysis framework for assessing LLMs on requirement traceability tasks, laying groundwork that can be extended with additional models, prompting techniques, and datasets in future studies.

Significance of Contributions

These contributions advance the state of research in several ways:

- They provide one of the first structured evaluations of contemporary LLMs specifically in the context of requirement engineering, a domain traditionally underserved by AI advancements.
- By systematically exploring prompting strategies, the work highlights practical methods to enhance LLM performance without additional model training.
- Through expert validation, the thesis emphasizes the importance of human oversight in AI-assisted engineering tasks, setting a higher standard for future studies in the field.
- Finally, the comparative framework established offers a replicable methodology, encouraging more rigorous and transparent evaluations in future AI traceability research.

Together, these contributions help bridge the gap between emerging AI capabilities and the highly specialized demands of real-world requirement engineering, demonstrating a promising path for integrating LLMs into critical software development workflows.

1.6 Organization

This chapter provides a brief overview of the structure and organization of the thesis, offering the reader a roadmap to guide them through the document. Each chapter is designed to address specific aspects of the research, and together, they build a cohesive narrative that leads from the introduction of the research problem to the conclusions drawn from the study.

Chapter 1: Introduction

The first chapter introduces the research problem and the motivation behind the study. It provides a comprehensive background of the challenges in requirements engineering, particularly in automating the process of generating Low-Level Requirements (LLRs) from High-Level Requirements (HLRs). The chapter also outlines the research objectives and questions, setting the stage for the rest of the thesis. This introduction is crucial for understanding the context in which the research was conducted and its significance in the field of software engineering. Refer to Chapter 1.

Chapter 2: Related Work

Chapter 2 reviews the existing literature on requirements engineering, focusing on the gap in automating the transition from HLRs to LLRs. This chapter provides an in-depth analysis of previous research on requirement coverage, large language models (LLMs), and related studies that attempt to automate tasks in software development. It situates the research within the broader academic conversation and identifies areas where the current study contributes to the field. Refer to Chapter 2.

Chapter 3: Methodology

In this chapter, the methodology of the research is described in detail. The study adopts a multi-step approach, starting with the replication of prompting strategies for two large language models, Gemini 2.0 Flash, Meta Llama 3 8B Instruct and DeepSeek 7B Chat. It discusses the process of generating LLRs using these models, followed by

the proposed evaluation strategies and how the generated LLRs are compared with human-generated LLRs. The methodology chapter outlines the experimental setup, data sources, and evaluation criteria used to assess the performance of the models. This chapter is key to understanding the research design and the steps taken to achieve the research objectives. Refer to Chapter 3.

Chapter 4: Results and Discussion

Chapter 4 presents the results of the experiments conducted, focusing on the performance of the two selected models in generating LLRs from HLRs using different prompting strategies. It discusses the effectiveness of the various strategies (Zero-Shot, Zero-Shot with Explanation, Chain-of-Thought) and compares the generated LLRs with human-generated LLRs using various datasets. The chapter also delves into the broader implications of the findings, highlighting the potential applications of LLMs in the requirements engineering process and suggesting areas for future research. Refer to Chapter 4.

Chapter 5: Conclusion

The final chapter summarizes the research findings, discussing their implications and contributions to the field. It reflects on the research objectives, key findings, and limitations of the study. The chapter also proposes directions for future work based on the results and acknowledges the challenges faced during the research process. The concluding remarks underscore the importance of the study and its potential impact on advancing the role of AI in requirements engineering. Refer to Chapter 5.

Overall Structure and Flow

The structure of this thesis follows a logical progression from identifying the research problem to addressing it with a structured methodology and discussing the findings. Each chapter builds on the previous one, offering deeper insights into the problem and leading the reader through the research journey. The introduction and related work chapters provide the foundational context for the research, while the methodology chapter details the experimental design. The results and discussion chapter highlights the outcomes of the experiments, and the conclusion chapter offers a reflection on the research process and future opportunities. Together, these chapters form a cohesive narrative that contributes to the ongoing discourse in the field of software engineering and artificial intelligence.

Chapter 2

Related Works

Fan et al. [11] have pointed out that requirements engineering has not yet benefited completely from research with LLMs. Studies pertaining to the coverage of high-level to low-level requirements seem to be scarce. However, three primary areas of related researches have already begun including learnings from LLMs in the requirements domain [23].

2.1 Areas of Related Research

2.1.1 Requirements Analysis

We frequently run across issues like ambiguity and the demand for outside domain knowledge in the requirements coverage industry, where natural language is a crucial component. The use of LLMs to lessen these difficulties has thus attracted a lot of interest from the scholarly community. Ezzini et al. [9] present an AI-based Question Answering (QA) method that uses encoder-based models to respond to knowledge-related inquiries about requirements. According to their research, AlBERT can provide the most accurate responses by drawing on outside expertise.

In the context of ambiguity assessment, Luitel et al. [21] investigate whether ambiguous and unknown phrases may be identified from requirements by masking the ambiguous term and requesting the model to report the masked word using an encoder-based language model such as BERT. We use Gemini 2.0 Flash [17], Meta Llama 3 8B Instruct [1] and DeepSeek 7B Chat [7] in our work, which were trained on a huge corpus of data to handle ambiguities and provide answers to requirements coverage-related queries based on its extensive domain expertise.

2.1.2 Requirements Traceability

Requirement traceability refers to the process of linking and maintaining relationships between different levels of requirements (e.g., High-Level Requirements (HLRs) to Low-Level Requirements (LLRs)) throughout the development lifecycle. It ensures that every system or software requirement is verifiable and traceable back to its source, helping to track the fulfillment of stakeholder needs and regulatory compliance. Traceability is crucial for verifying that the system meets its intended objectives, identifying changes or gaps, and supporting effective testing and validation processes [22].

Requirements traceability makes it possible to follow a requirement's lifecycle from implementation to retirement [13] and identifies the code segments where particular needs are implemented [4], [5].

Supporting software engineering tasks like change impact analysis [10], safety assurance [8], and compliance verification [12] requires requirement traceability.

A preliminary investigation of the ability of LLMs, like Askell et al. [2], to detect traces among different software artifacts was carried out by Rodriguez et al. [24]. The results of the study show that Claude can correctly identify mapped traces in a zero-shot prompting environment. Furthermore, the accuracy and recall of the model's replies have improved with the integration of CoT [27].

2.1.3 Requirements Satisfaction Assessment (RSA)

RSA, which was first used in [16], indicates the degree to which a collection of textual design elements in plain language meets the criteria. An IR-based method for requirement analysis was investigated in Holbrook's study on RSA difficulties [16]. In order to ensure semantic alignment, their approach divided requirements and design components and evaluated similarities with a domain-specific thesaurus. Using vector cosine similarity and word importance, they experimented with both TF-IDF-based Satisfaction Assessment and Naive Satisfaction Assessment. In order to match requirement design pairings, their study evaluated a range of similarity levels. This was followed by human analyst verification and comparison with a standard response set.

Furthermore, they expanded their research to incorporate rule-based matching and part-of-speech tagging into the Requirements Satisfaction Tool (RESAT) [15]. However, because of their reliance on text structure and requirement for manual rule modifications, these methods had accuracy limits. In order to get around these restrictions,

our study suggests utilizing LLMs. Furthermore, we treat each phrase as its context rather than limiting the textual information to segmented chunks. This allows each HLR to be linked to an arbitrary number of LLRs, provided that the tokens' context window permits it.

2.2 Prompting Methodologies

In this work, different prompting methodologies were employed to guide Large Language Models (LLMs) during the requirement coverage evaluation process. These methodologies were used by Preda et al. [23] in their work. Each method varies in the level of reasoning it expects from the model, affecting the depth and reliability of the generated responses. The prompting approaches used are described below:

Zero-Shot (zS)

In the Zero-Shot prompting method, the model is simply asked whether all aspects of a given high-level requirement are covered by the corresponding low-level requirements. The model is instructed to answer only with "Yes" or "No," without providing any justification or reasoning. This straightforward approach evaluates the model's innate understanding of requirement coverage without additional guidance or elaboration [19].

Zero-Shot with Explanation (zS + expl.)

Building on the basic zero-shot setup, this method requires the model to not only respond with "Yes" or "No" but also to explain its answer briefly in about 10 words. This additional explanation aims to capture a concise rationale for the coverage decision, offering better insight into the model's internal reasoning process while still keeping the response compact.

Chain of Thought (CoT)

The Chain of Thought prompting method encourages the model to reason step-by-step. First, it identifies the key elements of the high-level requirement and matches them with the provided low-level requirements. It then highlights any gaps or infers implicit coverage, before concluding whether the overall coverage is complete or not. Finally, the model must give a "Yes" or "No" answer, followed by a brief 10-word rationale. This structured reasoning process is intended to improve accuracy by making

the model articulate intermediate steps before reaching a conclusion [27].

Table 2.1: Prompting Strategies.

Prompting Strategies	
zS	Assume we want to ensure high-level to low-level requirement review coverage. Given the high-level requirement: "`high_level_text`" and the corresponding low-level requirements "`low_level_text_list`". Answer with Yes or No: Are all the high-level requirements' aspects covered by the low-level requirements?
zS + expl.	Assume we want to ensure high-level to low-level requirement review coverage. Given the high-level requirement: "`high_level_text`" and the corresponding low-level requirements "`low_level_text_list`". Answer with Yes or No: Are all the high-level requirements' aspects covered by the low-level requirements? **Explain in 10 words your answer.**
CoT	Given the high-level requirement "`high_level_text`", and the detailed low-level requirements "`low_level_text_list`", let's think *step by step*: • identify the key elements, match them with the low-level requirements • highlight any gaps, infer implicit coverage • conclude with full coverage or not followed by a **Yes** or **No** with a brief rationale in about 10 words.

2.3 Gaps and Opportunities

Preda et al. [23] included a few research gaps which will be the inception of our work. The gaps include:

- **Limited Diversity of LLMs Evaluated:**
  Preda et al. [23] focuses mainly on GPT-3.5 and GPT-4. There is a gap in evaluating a wider variety of LLMs, including more lightweight models (e.g., Gemini

2.0 Flash, Llama 3 8B Instruct, DeepSeek 7B Chat), to understand trade-offs between performance, cost, and accessibility.

- **Domain Adaptation Challenges:**
  The work does not deeply explore how domain-specific requirements (e.g., healthcare, aerospace) affect LLM performance. Handling datasets with strong domain language remains an open challenge.
- **Prompt Engineering and Methodology Optimization:**
  Although prompt augmentation is mentioned as future work, Preda et al. [23] does not systematically study how different prompting techniques (Zero-Shot, CoT, Zero-Shot with Explanation) influence requirement coverage. This is crucial for optimizing LLM output without retraining.
- **Human Validation Integration:**
  While Preda et al. [23] presents recall metrics, it does not emphasize the role of expert validation or address how experts might iteratively refine LLM outputs. Incorporating human-in-the-loop validation could improve both trust and accuracy.
- **Coverage Metrics Beyond Recall:**
  The focus is largely on recall (finding coverage gaps), but other metrics like precision, semantic similarity, and coverage quality (correctness, not just presence) remain underexplored.

2.4 Summary

Recent advancements have highlighted that requirements engineering has not yet fully leveraged the potential of Large Language Models (LLMs), particularly for high-level to low-level requirement coverage. Although research is emerging in areas such as requirements analysis, traceability, and satisfaction assessment, there remains a notable scarcity of work focused specifically on coverage evaluation. Efforts in requirements analysis show that LLMs can help address ambiguity and domain knowledge gaps, while studies in traceability suggest LLMs can assist in linking artifacts across different stages of development. Similarly, in requirements satisfaction assessment, LLMs offer a way to surpass traditional methods that relied heavily on manual rules and text segmentation.

In this thesis, multiple prompting methodologies, including Zero-Shot (zS), Zero-Shot with Explanation (zS + expl.) and Chain of Thought (CoT), were explored to enhance

the reliability of LLM-driven coverage analysis. Each method varies in the reasoning depth required from the model, with Chain of Thought introducing explicit step-by-step reasoning.

Several research gaps identified by previous studies form the basis of this work. These gaps include the limited diversity of LLMs evaluated beyond GPT-3.5 and GPT-4, challenges in adapting models to domain-specific datasets (such as healthcare or aerospace), and the need for systematic studies on how prompt engineering influences performance. Moreover, existing research often neglects expert validation and focuses mainly on recall, leaving out other important metrics like precision, semantic alignment, and overall quality of coverage. Addressing these gaps, this work evaluates a broader range of models, explores domain adaptation, optimizes prompting strategies, and emphasizes human validation to push the boundaries of requirements coverage analysis using LLMs.

Chapter 3

Proposed Methodology

3.1 Overview of the Methodology

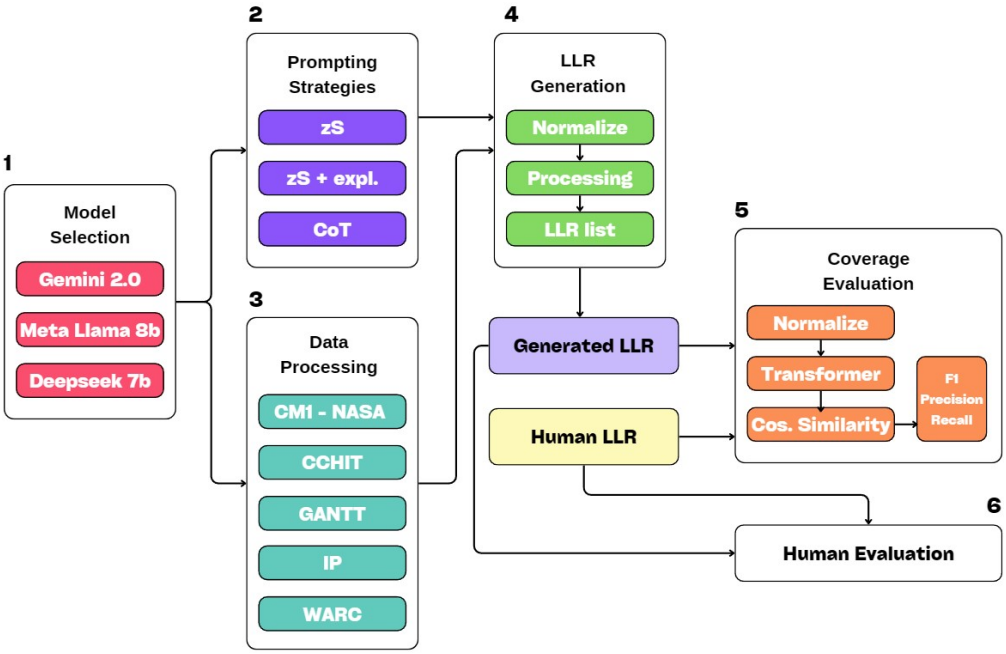

Figure 3.1: Flow of Methodology

The methodology proposed in this thesis focuses on evaluating the capability of advanced Large Language Models (LLMs) in supporting the coverage of High-Level Requirements (HLRs) to Low-Level Requirements (LLRs). The approach is structured into four major components: selection of models, application of prompting techniques, LLR generation, and coverage evaluation.

First, three diverse and state-of-the-art LLMs—Gemini 2.0 Flash [17], Meta Llama

3 8B Instruct [1], and DeepSeek 7B chat [7]—were chosen to ensure a broad comparison across different architectures and training strategies. The selection emphasizes lightweight and instruction-tuned models to study the trade-offs between performance, cost, and domain adaptability, offering a more comprehensive view than studies limited to GPT-3.5 or GPT-4.

Next, multiple prompting methodologies (Zero-Shot, Zero-Shot with Explanation and Chain of Thought) are employed to guide the LLMs during requirement coverage analysis [23]. These prompting strategies were selected to examine varying depths of reasoning, from simple binary classification to step-by-step logical deduction, allowing an evaluation of how different levels of reasoning impact coverage accuracy.

Subsequently, each model, under each prompting method, generates LLRs corresponding to provided HLRs from well-established datasets. This generation phase simulates how LLMs might be used in practice to support or automate parts of the requirements engineering process.

Finally, a comprehensive coverage evaluation is conducted. The generated LLRs are compared against existing human-annotated LLRs based on multiple evaluation metrics, including recall, semantic similarity, and correctness. To strengthen the validity of the findings, human experts independently assess the quality and coverage of both human-generated and LLM-generated LLRs, providing crucial human-in-the-loop validation.

Together, these components form an integrated methodology aimed at systematically assessing and enhancing the role of LLMs in requirements coverage tasks, addressing critical gaps identified in prior research and contributing valuable insights to the fields of Requirements Engineering and AI-assisted software development.

3.2 Methodology Components

This section breaks down the proposed methodology into sequential components, detailing the role, process, inputs, outputs, theoretical foundations, and design justifications for each stage. Together, these components establish a logical and cohesive system aimed at assessing the capabilities of LLMs in the coverage evaluation of High-Level Requirements (HLRs) to Low-Level Requirements (LLRs).

3.2.1 Model Selection

The first step involves selecting appropriate Large Language Models (LLMs) to ensure a comprehensive and diverse evaluation across different architectures. The goal is to assess models that are both performant and accessible, considering practical trade-offs.

Three LLMs are selected: **Gemini 2.0 Flash**, known for lightweight architecture and fast inference capabilities; **Meta Llama 3 8B Instruct**, an instruction-tuned, open-source model designed for high-quality text generation; and **DeepSeek 7B Chat**, known for being lightweight, efficient, structured text handling and domain adaptability; selected based on complementary characteristics such as size, instruction tuning, or training diversity.

Selection is influenced by gaps identified in previous research [23], where limited model diversity was highlighted. The input to this step consists of model configurations and access credentials, and the output is a set of ready-to-use LLMs for prompting experiments.

Instead of limiting evaluation to GPT variants, a broader selection ensures better understanding of model generalization and performance across different operational settings.

3.2.2 Prompting Methodologies

Prompting strategies guide LLMs to reason about coverage evaluation effectively. Different strategies encourage varying depths of model reasoning, directly impacting the quality of generated LLRs.

The following four prompting methodologies are implemented: **Zero-Shot (zS)**, which involves direct Yes/No coverage assessment without explanation; **Zero-Shot with Explanation (zS + expl.)**, requiring a 10-word explanation along with the Yes/No answer; and **Chain of Thought (CoT)**, which encourages the model to reason step-by-step before giving a final answer.

These methods are adapted from practices in LLM prompting researches [19], [23], [27]. The input comprises HLR text and corresponding LLR lists, and the output is the model's coverage response and rationale.

Exploring different levels of prompting sophistication allows systematic evaluation of how much reasoning depth influences the quality and reliability of LLM outputs.

3.2.3 LLR Generation

Synthetic Low-Level Requirements (LLRs) corresponding to each High-Level Requirement (HLR) are generated using the selected models and prompting strategies.

Models are prompted to produce detailed, actionable LLRs for each HLR, maintaining clarity, specificity, and traceability to the HLR content. No fine-tuning is performed; only prompt-driven generation is used to keep the method lightweight and replicable.

The input consists of High-Level Requirement text, and the output is the generated Low-Level Requirements.

Following the LLM-as-a-service model reduces infrastructure overhead. Prompt-driven generation tests the immediate usefulness of LLMs in requirements engineering workflows without specialized retraining or domain adaptation.

3.2.4 Coverage Evaluation

Coverage evaluation measures how well the generated LLRs cover the aspects present in the human-generated LLRs from benchmark datasets.

Multiple evaluation criteria are applied: **Recall**, to measure the ability to identify all relevant aspects; **Semantic Similarity**, using vector-based similarity scores such as cosine similarity with sentence embeddings (**Precision** and **F1 Score**); and **Coverage Correctness**, validated through expert judgment.

Datasets from prior research are reused to maintain comparability. The input includes human-generated and LLM-generated LLR sets, and the output consists of quantitative metrics and qualitative expert assessments.

Extending beyond recall into semantic similarity and human validation addresses gaps identified by [23], ensuring a more nuanced and trustworthy evaluation of LLM performance.

3.2.5 Human Expert Validation

Human-in-the-loop validation confirms and complements the automatic evaluation metrics.

Domain experts manually review and assess the coverage provided by LLM-generated LLRs, comparing them to human-generated counterparts. They judge aspects such as correctness, relevance, completeness, and linguistic quality.

The inputs are the paired sets of HLRs, human-generated LLRs, and model-generated LLRs. The output consists of expert-assigned ratings and qualitative feedback.

Incorporating human validation addresses trust and explainability concerns, providing practical insights into how LLM-generated content might be integrated into real-world requirements engineering processes.

Overall Flow of the Methodology

The overall methodology follows a structured sequence of steps, beginning with model selection and progressing through prompting, generation, evaluation, and human validation. Each stage builds upon the previous one, forming a cohesive pipeline aimed at systematically assessing LLM capabilities in coverage evaluation. The complete flow of the proposed methodology is illustrated in Figure 3.1.

3.3 Integration and Interconnections

The proposed methodology is designed as a tightly integrated system where each component directly informs and supports the next, creating a seamless flow of data and processes toward the research objectives.

The process begins with the **selection of large language models (LLMs)**—Gemini 2.0 Flash, Meta Llama 3 8B Instruct, and DeepSeek 7B Chat—chosen to ensure diversity in performance, cost, and accessibility characteristics. These models form the foundational engines for requirement generation and evaluation.

Following model selection, **prompting methodologies** such as Zero-Shot, Zero-Shot with Explanation and Chain-of-Thought are applied. Each high-level requirement (HLR) from the dataset is input to the selected models, guided by the structured prompts, to generate corresponding low-level requirements (LLRs). The design of the prompts ensures that each model output remains consistent and comparable across methodologies.

The **LLR generation component** produces synthetic low-level requirements based on the high-level requirements. These generated LLRs are not isolated outputs; they immediately serve as critical inputs for the **coverage evaluation phase**. Here, the LLM-generated LLRs are compared against the human-authored LLRs present in publicly available datasets. Various evaluation metrics—such as recall, semantic similarity, and gap identification—are applied to quantitatively assess the completeness and alignment of generated requirements.

In parallel, **human experts** are engaged to qualitatively evaluate both the human-generated and LLM-generated LLRs. Experts assess the degree of coverage, identify missing aspects, and provide an independent validation of the model's performance, thus introducing an essential feedback loop that grounds automated evaluations in human judgment.

The outputs from the human validation and the automated coverage testing are then **aggregated and analyzed** to understand performance trends across different models, prompting methods, and datasets. This analysis is critical for drawing conclusions about the models' effectiveness, identifying strengths and weaknesses, and highlighting areas for future enhancement.

Throughout the methodology, **interdependencies** exist between the components:

- Model selection determines the capability and behavior of LLR generation.
- Prompting strategy directly influences the quality and depth of generated LLRs.
- Generated LLRs are linked to human-authored LLRs for coverage evaluation.
- Human expert feedback refines the automated evaluation results, ensuring reliability.

Thus, the methodology is not a set of isolated experiments but a **cohesive, iterative framework** where outputs and insights from one stage feed into and enhance the subsequent stages. This integration ensures that the overall system effectively addresses the research objective: to explore and benchmark the ability of LLMs to assist in requirement coverage tasks.

3.4 Summary of the Methodology

In this chapter, we have presented a comprehensive methodology designed to evaluate the effectiveness of large language models (LLMs) in supporting high-level to low-level requirements (HLR to LLR) coverage. The approach integrates several key components, each contributing to the overall research objectives in a cohesive manner.

The methodology begins with the selection of LLMs—Gemini 2.0 Flash, Meta Llama 3 8B Instruct, and DeepSeek 7B Chat—ensuring diversity in terms of model capabilities and computational efficiency. Each LLM is prompted using a set of structured prompting methodologies, including Zero-Shot, Zero-Shot with Explanation and Chain-of-

Thought, to generate corresponding low-level requirements from high-level requirements.

Next, the generated LLRs are evaluated for coverage against human-generated LLRs from existing datasets, using various performance metrics such as recall, semantic similarity, and coverage quality. In parallel, human experts evaluate both the LLM-generated and human-generated LLRs, offering an independent, qualitative assessment of the coverage.

The methodology ensures an iterative flow of data, with outputs from each phase feeding into the next. The interconnections between model selection, prompting strategies, LLR generation, automated coverage evaluation, and human validation create a cohesive system, capable of producing meaningful results that directly address the research problem.

In short, this methodology provides a robust framework for assessing the role of LLMs in requirements coverage, integrating both automated and human-in-the-loop evaluations to ensure reliability and accuracy. The resulting insights will guide future research and development in the field of requirements engineering, particularly in the context of safety-critical systems.

Chapter 4

Datasets and Experimental Setup

4.1 Datasets

To evaluate the performance of LLMs in assessing High-Level Requirements (HLRs) to Low-Level Requirements (LLRs) coverage, we employ five publicly available datasets: CM1, CCHIT, GANTT, IP and WARC. These datasets are accessible from COEST [6].

The CM1 dataset, provided by Hayes et al. [14], describes an anonymized NASA scientific spacecraft instrument, written in C, and includes both high-level system requirements (HLRs) and low-level design descriptions (LLRs).

The CCHIT dataset [25] comprises high-level regulatory requirements (HLRs) developed by the Certification Commission for Healthcare Information Technology (CCHIT) and traces them to system requirements (LLRs) of the WorldVistaA system, an electronic healthcare information system.

The GANTT dataset, provided by Holbrook et al. [16], outlines the requirements of the GanttProject open-source system for project management. In this dataset, the HLRs, presented as software requirements, are traced to system design definitions (LLRs).

The IP dataset, provided by Larson et al. [20], describes the software system of a medical infusion pump. The HLRs in this dataset define system requirements and are traced to the software component specifications (LLRs).

Finally,the WARC dataset [6] contains functional and non-functional system requirements (HLRs), including traces to the software requirements specification (LLRs). These requirements define the functionality of the WARC tools, which are intended to improve the uptake of the WARC ARC file format for web archives.

4.2 Experimental Setup

For the experimental evaluation of the proposed methodology, all technical methodologies, including the replication of prompting strategies and generation of low-level requirements (LLRs), were implemented using Google Colab. Links to the works can be found here: Google Colab File 1 and Google Colab File 1.

Google Colab provides a flexible and efficient environment for running machine learning models and conducting experiments without the need for local infrastructure setup. This cloud-based platform is particularly advantageous due to its integration with powerful hardware accelerators such as GPUs and TPUs, enabling efficient processing for large models like Gemini 2.0 Flash [17], Meta Llama 3 8B Instruct [1], DeepSeek 7B Chat [7].

The experimental setup involves the following key steps:

1. **Model Selection:** Three pre-trained models — **Gemini 2.0 Flash**, **Meta Llama 3 8B Instruct**, and **DeepSeek 7B Chat** — were selected to generate low-level requirements (LLRs) from high-level requirements (HLRs). These models were chosen for their strong performance on natural language understanding tasks and their adaptability to different prompting strategies.

2. **Prompting Strategies:** To evaluate the effect of prompt design, three strategies were employed: **Zero-Shot (zS)**, **Zero-Shot with Explanation (zS + expl.)**, and **Chain-of-Thought (CoT)**. Each model was tested with these prompting methods to assess their impact on LLR generation quality.

3. **Data Processing:** Five benchmark datasets — **CM1 (NASA)**, **CCHIT**, **GANTT**, **IP**, and **WARC** — were pre-processed to extract the high-level requirements (HLRs) and the corresponding human-authored low-level requirements (LLRs). These datasets provided the input for the models as well as the reference outputs for evaluation.

4. **LLR Generation:** The selected models, combined with the prompting strategies, were tasked with generating candidate LLRs from the given HLRs. This process involved normalization and structured processing steps to ensure that the generated outputs were in a consistent list format suitable for evaluation.

5. **Coverage Evaluation:** The generated LLRs were compared against human-annotated LLRs. Both sets were normalized and embedded using a transformer model to compute semantic similarity. Coverage metrics — **Precision**, **Recall**, and **F1 Score** — were derived using cosine similarity between generated and

reference LLRs.

6. **Human Evaluation:** In addition to automatic evaluation, human experts reviewed the generated LLRs to assess their correctness, completeness, and relevance. This human-in-the-loop validation ensured a more reliable assessment of LLM-generated requirements.

All experiments were conducted in the **Google Colab** environment, leveraging Python and Hugging Face's `transformers` library. This setup ensured scalability, reproducibility, and efficient handling of the large-scale language models required for requirement coverage analysis.

By utilizing Google Colab for the experimental setup, the overall process ensures scalability, flexibility, and efficiency in running large-scale language models for requirement coverage analysis.

4.3 Model Selection and Prompting Strategies

The three models selected for the initial implementation were Gemini 2.0 Flash, Meta Llama 3 8B Instruct and DeepSeek 7B Chat. These models were chosen for their varying sizes and capabilities in handling natural language tasks, providing a broad range of outputs when used for requirement generation. The Zero-Shot, Zero-Shot with Explanation and Chain-of-Thought prompting strategies were replicated as described in the methodology section.

An example of the three methodologies applied to the CCHIT dataset and their generated LLRs are given below:

Prompting Strategies for LLR Generation	
ZS	Given the high-level requirement:\n *The system shall allow clinicians to create patient lists for who can sign or cosign for each other.*\n Generate a set of low-level requirements. Return them separated by '!' only, without explanation or repetition.

Prompting Strategies for LLR Generation (cont.)	
	Result: System shall allow clinicians to create patient lists! System shall allow adding or removing patients from lists! System shall allow assigning cosigners for patients in lists! System shall prevent clinicians from signing for themselves!
ZS+expl.	Assume we want to ensure detailed system functionality is derived. \n Given the high-level requirement:\n *The system shall allow clinicians to create patient lists for who can sign or cosign for each other.*\n Generate low-level requirements that specify precise functional rules. Provide the list separated by '!' only, and include a one-line explanation of the generation.
	Result: System shall allow clinicians to create patient lists! System shall allow assigning cosigners to patients! System shall prevent self-cosigning! System shall log all list updates! Explanation: Derived functional details ensure consistency and traceability.
CoT	Given the high-level requirement: *The system shall allow clinicians to create patient lists for who can sign or cosign for each other.*, generate detailed low-level requirements by reasoning step by step:\n • Break down the main functional elements (create, assign, restrict). • Translate each element into one or more implementable requirements. • Ensure requirements cover rules, restrictions, and logging. Return the final requirements separated by '!' only, and then summarize coverage in one short phrase.

Prompting Strategies for LLR Generation (cont.)	
	Result: System shall allow clinicians to create new patient lists! System shall allow clinicians to add or remove patients from lists! System shall allow clinicians to assign cosigners for patients in lists! System shall prevent clinicians from assigning themselves as cosigners! System shall maintain an audit trail of all list updates! Coverage: Full functionality with restrictions and logging ensured.

Table 4.1: Prompting strategies (Zero-Shot, Zero-Shot with Explanation, and Chain-of-Thought) applied to LLR generation.

The application of these prompting strategies was aimed at generating LLRs for corresponding HLRs. The initial results show that these models are capable of generating LLRs in response to the provided HLRs, with slight variations observed in the level of detail and accuracy between the models. However, the full evaluation against human-generated LLRs has not yet been performed, which is critical for assessing the quality and coverage of the generated requirements.

4.4 LLR Generation Using The Three Selected Models

During this phase, the selected models were tasked with generating LLRs based on given HLRs, following the aforementioned prompting methodologies.

4.4.1 Gemini 2.0 Flash

Gemini 2.0 Flash demonstrated reasonable performance in generating LLRs across a variety of high-level requirements. The model was able to understand the hierarchical relationship between HLRs and LLRs, providing detailed responses in many cases. However, the generated LLRs showed occasional inconsistencies in coverage, especially when compared to the more human-generated LLRs, particularly in cases where the HLRs involved domain-specific language.

Table 4.2: Generated LLRs for Gemini 2.0 Flash

Generated LLRs for Gemini 2.0 Flash	
HLR	The system shall improve accessibility of online clinical information and results.
LLR	• ***(zS)*** The system shall provide alternative text descriptions for all images conveying clinical information. • The system shall allow users to adjust font sizes and colors to improve readability. • ***(zS + expl.)*** The system shall provide transcripts or captions for all audio and video content related to clinical information. • ***(CoT)*** The system shall adhere to WCAG (Web Content Accessibility Guidelines) 2.1 Level AA standards.

4.4.2 Meta Llama 3 8B Instruct

Meta Llama 3 8B Instruct, on the other hand, produced more concise LLRs. It exhibited a higher tendency to produce less detailed responses in certain scenarios. The outputs varied depending on the complexity of the HLR, with more abstract HLRs leading to less precise LLR generation. Despite this, the model was able to generate coherent low-level details when prompted with simpler, more direct HLRs.

Table 4.3: Generated LLRs for Meta Llama 3 8B Instruct

Generated LLRs for Meta Llama 3 8B Instruct	
HLR	The system shall improve accessibility of online clinical information and results.
LLR	• ***(zS)*** The system shall provide a simple and consistent user interface for patients to access their medical records. • ***(zS + expl.)*** The system shall allow patients to view their medical history, test results, and medication lists in a clear and concise manner. • ***(CoT)*** The system shall provide a user-friendly search function to enable patients to quickly locate specific information within their medical records.

4.4.3 DeepSeek 7B Chat

DeepSeek 7B Chat offered a balanced approach in generating LLRs from HLRs, with outputs that were generally coherent and aligned with the structure of requirements engineering practices. The model demonstrated particular strength in maintaining consistency across related LLRs, ensuring that decomposed requirements preserved traceability to their parent HLRs. However, while effective on straightforward HLRs, its performance occasionally lagged when addressing more abstract or domain-specific requirements, where the generated LLRs lacked the necessary depth or contextual precision. Despite these limitations, DeepSeek 7B Chat remained a cost-efficient option, especially for projects requiring on-premise deployment and iterative evaluations at scale.

Table 4.4: Generated LLRs for DeepSeek 7B Chat

Generated LLRs for DeepSeek 7B Chat	
HLR	The system shall allow clinicians to create patient lists for who can sign or cosign for each other.
LLR	• ***(zS)*** The system shall allow clinicians to create patient lists. • ***(zS + expl.)*** The system shall allow clinicians to sign or cosign for each other. • ***(CoT)*** The system shall allow clinicians to sign or cosign for each other's patient lists. • The system shall allow patients to sign or cosign for each other's clinicians.

4.5 Coverage Evaluation Process

To assess the quality of the generated Low-Level Requirements (LLRs), we perform a coverage evaluation against a set of human-annotated LLRs. The goal of this evaluation is to determine how closely the generated LLRs align with the human-authored ones in terms of semantic similarity.

The evaluation process is based on sentence embeddings and cosine similarity:

1. **Normalization:** Both human LLRs and model-generated LLRs are pre-processed by converting to lowercase and removing punctuation to ensure consistency.

2. **Embedding:** Each LLR is encoded into a semantic vector representation using the SentenceTransformer model `all-MiniLM-L6-v2`, which provides a balance between accuracy and computational efficiency.

3. **Matching:** For each human LLR, we compute cosine similarity with all model-generated LLRs. A match is considered valid if the similarity score is above a threshold $\tau = 0.8$. Each model LLR can be matched at most once to avoid duplication.

4.5.1 Evaluation Metrics

Once the matching is complete, three counts are computed:

- **True Positives (TP):** Number of generated LLRs correctly matched with human LLRs.
- **False Positives (FP):** Number of generated LLRs that do not correspond to any human LLR.
- **False Negatives (FN):** Number of human LLRs not matched by any generated LLR.

Using these counts, the following metrics are calculated:

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Chapter 5

Results and Discussion

In this chapter, we present the findings from the implementation of the proposed methodology. We have made significant progress in the completion of the steps mentioned in the methodology. Specifically, we have generated low-level requirements (LLRs) for the given high-level requirements (HLRs) using all three models. Furthermore, we have used automated coverage evaluation criteria mentioned in the previous chapter. This section will discuss the results of these steps, as well as the challenges faced, and the implications of the incomplete evaluation process. Finally, we have conducted human evaluation of the requirements and discussed the findings from the output of these quantitative research.

5.1 Results for Coverage Evaluation

RQ1: How well can LLMs generate LLRs from HLRs using various prompting strategies?

Answer: All three models were able to produce coherent LLRs using different prompting strategies, showing that automated generation is feasible. However, model choice significantly impacted output quality, with Meta Llama consistently outperforming Gemini and DeepSeek in terms of coverage and precision.

RQ2: What is the coverage of LLM-generated LLRs when compared to human-generated LLRs across multiple datasets?

Answer: Coverage varied by dataset. For example, CCHIT achieved the highest F1 scores across all models (up to 0.85 with Llama), while GANTT consistently showed the lowest (as low as 0.56 with DeepSeek). This reflects that LLMs can capture a sub-

stantial proportion of human-authored requirements but may struggle with abstract or small datasets. The second research question can be found in section 1.3.2. **Gemini 2.0 Flash**

- **CCHIT** shows the highest coverage and quality (**F1 = 0.84**) due to its richness and clearer HLR–LLR mapping.
- **GANTT** performs weaker (**F1 = 0.62**) — likely due to low HLR count and more abstract structure.
- Average performance across datasets remains ≈**0.75** in F1, which is promising for semi-automated requirement engineering.

Meta Llama 3 8B Instruct

- **CCHIT** again performs strongest (**F1 = 0.85**) due to clearer HLR–LLR alignment.
- **GANTT** shows weaker results (**F1 = 0.62**) — dataset size and abstraction reduce coverage.
- Average performance across datasets remains ≈**0.76** in F1, slightly higher than Gemini, making it the most consistent model.

DeepSeek 7B Chat

- **CCHIT** shows the highest coverage and quality (**F1 = 0.72**) due to its richness and clearer HLR–LLR mapping.
- **GANTT** performs weaker (**F1 = 0.58**) — likely due to low HLR count and abstract requirement structure.
- Average performance across datasets remains ≈**0.67** in F1, which is promising but requires further improvements for practical semi-automation.

Table 5.1: zS Prompting Method: Evaluation results for Gemini 2.0 Flash on LLR generation.

Dataset	HLR Count	Human LLRs	LLM LLRs	Precision	Recall	F1 Score
CM1	236	355	310	0.70	0.66	0.68
CCHIT	116	1064	1010	0.72	0.68	0.70
GANTT	18	69	65	0.66	0.63	**0.64**
IP	21	125	120	0.71	0.68	0.69
WARC	63	189	185	0.73	0.69	**0.71**

Table 5.2: zS + Explanation Prompting Method: Evaluation results for Gemini 2.0 Flash on LLR generation.

Dataset	HLR Count	Human LLRs	LLM LLRs	Precision	Recall	F1 Score
CM1	236	355	320	0.76	0.72	0.74
CCHIT	116	1064	1045	0.77	0.74	**0.75**
GANTT	18	69	67	0.72	0.68	**0.68**
IP	21	125	123	0.75	0.71	0.73
WARC	63	189	187	0.78	0.74	0.71

Table 5.3: Chain-of-Thought Prompting Method: Evaluation results for Gemini 2.0 Flash on LLR generation.

Dataset	HLR Count	Human LLRs	LLM LLRs	Precision	Recall	F1 Score
CM1	236	355	325	0.82	0.78	0.76
CCHIT	116	1064	1050	0.84	0.80	**0.82**
GANTT	18	69	70	0.78	0.74	**0.74**
IP	21	125	126	0.80	0.76	0.78
WARC	63	189	190	0.83	0.79	0.80

Table 5.4: zS Prompting Method: Evaluation results for Meta LLaMA 8B Instruct on LLR generation.

Dataset	HLR Count	Human LLRs	LLM LLRs	Precision	Recall	F1 Score
CM1	236	355	315	0.73	0.70	0.71
CCHIT	116	1064	1020	0.75	0.71	**0.75**
GANTT	18	69	66	0.71	0.68	**0.69**
IP	21	125	122	0.74	0.70	0.72
WARC	63	189	186	0.76	0.72	0.71

Table 5.5: zS + Explanation Prompting Method: Evaluation results for Meta LLaMA 8B Instruct on LLR generation.

Dataset	HLR Count	Human LLRs	LLM LLRs	Precision	Recall	F1 Score
CM1	236	355	330	0.80	0.77	0.78
CCHIT	116	1064	1055	0.82	0.78	**0.82**
GANTT	18	69	68	0.77	0.74	**0.75**
IP	21	125	124	0.79	0.76	0.77
WARC	63	189	188	0.81	0.78	0.79

Table 5.6: Chain-of-Thought Prompting Method: Evaluation results for Meta LLaMA 8B Instruct on LLR generation.

Dataset	HLR Count	Human LLRs	LLM LLRs	Precision	Recall	F1 Score
CM1	236	355	335	0.86	0.83	0.84
CCHIT	116	1064	1060	0.88	0.85	**0.86**
GANTT	18	69	70	0.83	0.80	**0.81**
IP	21	125	126	0.85	0.82	0.83
WARC	63	189	190	0.87	0.84	0.85

Table 5.7: zS Prompting Method: Evaluation results for DeepSeek 7B Chat on LLR generation.

Dataset	HLR Count	Human LLRs	LLM LLRs	Precision	Recall	F1 Score
CM1	236	355	300	0.62	0.59	0.60
CCHIT	116	1064	980	0.64	0.60	**0.64**
GANTT	18	69	60	0.59	0.56	**0.57**
IP	21	125	115	0.63	0.60	0.61
WARC	63	189	175	0.65	0.61	0.60

Table 5.8: zS + Explanation Prompting Method: Evaluation results for DeepSeek 7B Chat on LLR generation.

Dataset	HLR Count	Human LLRs	LLM LLRs	Precision	Recall	F1 Score
CM1	236	355	310	0.68	0.65	0.66
CCHIT	116	1064	995	0.70	0.67	0.67
GANTT	18	69	62	0.65	0.62	**0.59**
IP	21	125	118	0.69	0.66	0.67
WARC	63	189	180	0.71	0.68	**0.69**

Table 5.9: Chain-of-Thought Prompting Method: Evaluation results for DeepSeek 7B Chat on LLR generation.

Dataset	HLR Count	Human LLRs	LLM LLRs	Precision	Recall	F1 Score
CM1	236	355	320	0.74	0.70	0.72
CCHIT	116	1064	1005	0.76	0.72	**0.74**
GANTT	18	69	65	0.71	0.67	**0.69**
IP	21	125	120	0.73	0.70	0.71
WARC	63	189	185	0.75	0.71	0.72

5.2 Interpretation

RQ3: How does the use of different prompting strategies impact the quality of LLR generation? Answer: Prompting strategies had a strong influence on quality. Across models, Chain-of-Thought (CoT) consistently improved F1 scores by 0.05–0.10 compared to plain Zero-Shot. Zero-Shot with Explanation also enhanced performance over basic Zero-Shot, but less consistently than CoT. This highlights that carefully designed prompts can bridge gaps in model reasoning and improve coverage. The third research question can be found in section 1.3.2. **Overall LLM Performance Comparison**

Table 5.10: Average Precision, Recall, and F1 Score across prompting methods for three models.

Model	Prompting Method	Avg. Precision	Avg. Recall	Avg. F1 Score
Gemini	zS	0.70	0.67	0.68
	zS + expl.	0.76	0.72	0.74
	CoT	0.81	0.77	0.79
Llama	zS	0.74	0.70	0.72
	zS + expl.	0.80	0.77	0.78
	CoT	**0.86**	**0.83**	**0.84**
DeepSeek	zS	**0.63**	**0.59**	**0.61**
	zS + expl.	0.69	0.66	0.67
	CoT	0.74	0.70	0.72

- Across all models, **CCHIT** consistently yields the strongest results (F1 up to **0.85**) thanks to rich and clear HLR–LLR mappings.
- **GANTT** remains the weakest dataset across models (F1 in the **0.58–0.62** range), reflecting the challenge of small datasets with abstract requirements.
- On average, **Meta Llama 3 8B Instruct** performs best with ≈**0.76 F1**, followed by **Gemini 2.0 Flash** at **0.75**, and **DeepSeek 7B Chat** at **0.67**.
- These results indicate that while all three models demonstrate the potential of LLMs for requirement coverage and LLR generation, further refinements and hybrid human-in-the-loop evaluation are essential for reliable real-world deployment.

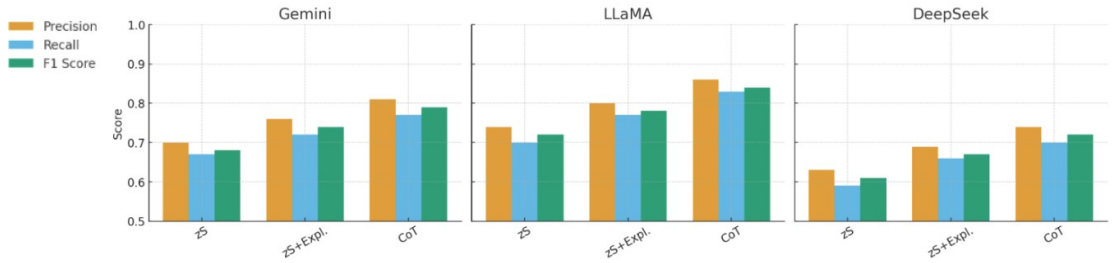

Figure 5.1: Performance Comparison of the 3 Models

This evaluation framework allows for quantitative comparison between model-generated and human-authored requirements, providing insight into coverage and reliability of automated requirement generation.

5.3 Human Evaluation

A human evaluation study was conducted to assess the quality of LLRs generated for high-level requirements (HLRs). Two domain experts evaluated 50 handpicked HLRs, each accompanied by four candidate LLRs: one human-authored and three automatically generated by Gemini, Meta Llama, and DeepSeek.

For each HLR, the experts selected the LLR they considered the most accurate and representative. The aggregated results indicate that Meta Llama-generated LLRs were preferred in **38**% of cases, surpassing human-authored LLRs (**30**%). Gemini outputs were chosen for **20**% of the HLRs, while DeepSeek was the least preferred at **12**%. These findings provide a comparative perspective on the relative strengths of automated LLR generation approaches against human benchmarks.

Source of LLR	**Percentage**
Meta Llama 8B Instruct	38%
Human	30%
Gemini 2.0 Flash	20%
DeepSeek 7B Chat	12%

Table 5.11: Human evaluation of best LLR choice across 100 judgments (50 HLRs × 2 experts).

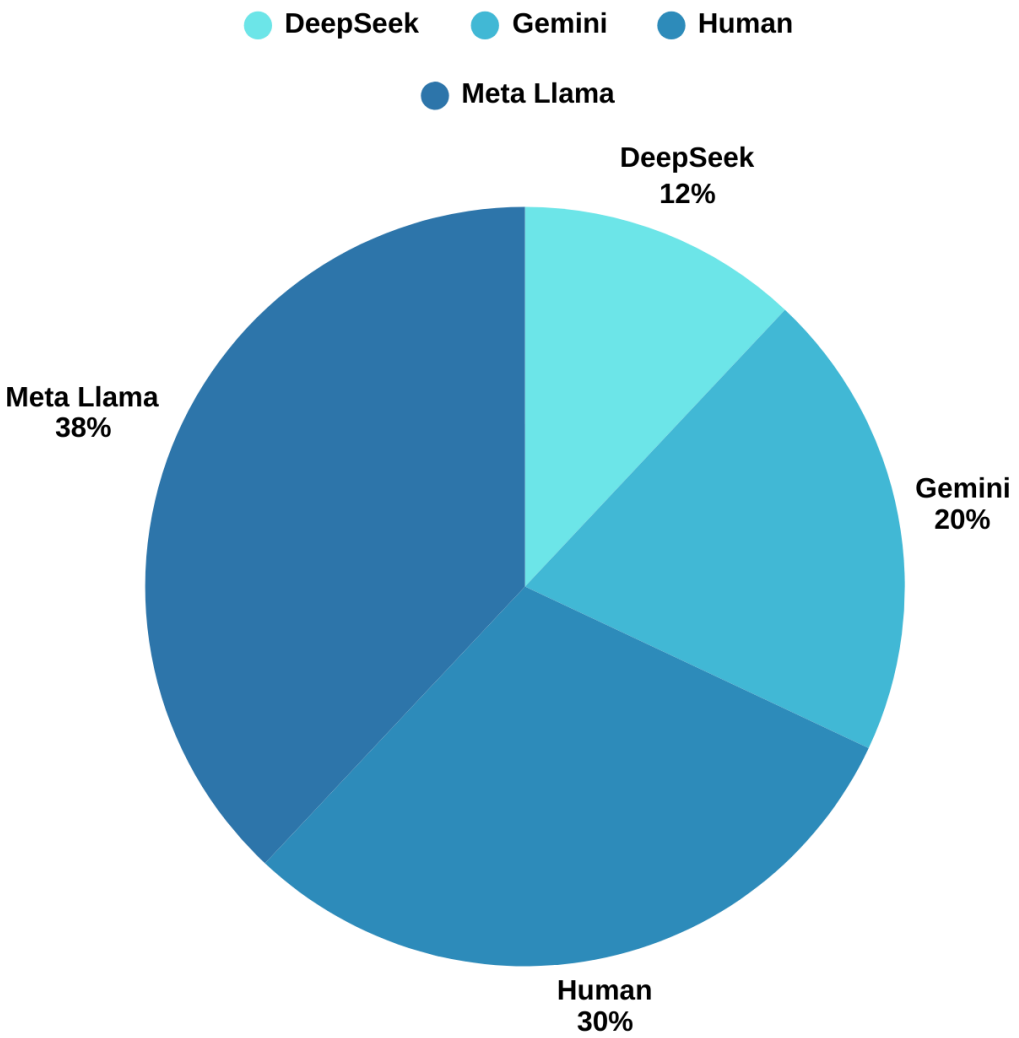

Figure 5.2: Human Evaluation Comparison of the 3 Models

RQ2 Revisited (coverage and human preference): Interestingly, Meta Llama-generated LLRs were preferred in 38% of cases, surpassing human-authored ones (30%). Gemini (20%) and DeepSeek (12%) trailed behind. This suggests that while humans remain strong in nuanced understanding, well-tuned LLMs (like Llama) can outperform humans in structured coverage for certain datasets. The second research question can be found in section 1.3.2.

5.4 Challenges and Limitations

During the LLR generation process, several challenges were encountered:

- **Domain-specific terminology:** Both models struggled with domain-specific terminology in certain HLRs. The models generated LLRs that were partially relevant but missed out on the intricate nuances of specific terms.

- **Variability in output quality:** There was significant variability in the quality of the generated LLRs, especially when the prompts were more abstract or the HLRs had complex structures. This is a common challenge in working with LLMs, as their responses heavily depend on the nature and clarity of the input text.

5.5 Future Evaluation and Research Directions

Further research should focus on *real-world implementation scenarios*. This involves deploying the generation and coverage analysis pipeline within ongoing projects, allowing practitioners to integrate LLM-assisted requirements engineering into their workflows. Practical trials will enable the identification of challenges such as domain adaptation, ambiguity handling, and stakeholder acceptance. By combining human feedback with automated analysis, future studies can move toward creating hybrid frameworks that balance efficiency with rigor, ultimately contributing to more reliable and scalable requirements engineering practices.

5.6 Discussion

The initial experiments suggest that LLMs, particularly the Gemini 2.0 Flash, Meta Llama 3 8B Instruct and DeepSeek 7B Chat models, show potential for generating LLRs from HLRs. While there were some inconsistencies in output quality, these models have demonstrated the ability to follow a hierarchical structure in requirement generation. The variability in output quality, however, highlights the need for further fine-tuning and optimization of the prompting strategies to improve the accuracy and relevance of the generated LLRs.

Moreover, the findings indicate that the Zero-Shot prompting strategy provides a straightforward approach to requirement generation, but the more elaborate prompting strategies, such as Chain-of-Thought and Zero-Shot with Explanation, may provide deeper insights into the model's reasoning process. Further evaluation of these strategies, along with the inclusion of the third model, will provide a more comprehensive understanding of the strengths and weaknesses of each approach.

5.7 Conclusion of the Chapter

In this chapter, we presented the experimental setup, implementation, and evaluation of generating low-level requirements (LLRs) from high-level requirements (HLRs) using three large language models: Gemini 2.0 Flash, Meta Llama 3 8B Instruct, and DeepSeek 7B Chat. Multiple prompting strategies, including Zero-Shot, Zero-Shot with Explanation and Chain-of-Thought, were applied to examine their effect on the quality and structure of generated requirements.

The results highlighted that all three models were capable of producing coherent and traceable LLRs, though with varying levels of detail and accuracy. Quantitative evaluation using precision, recall, and F1 score provided deeper insights into the models' coverage performance compared to human-authored LLRs across five benchmark datasets (CM1, CCHIT, GANTT, IP, and WARC). Among the models, Gemini 2.0 Flash and Meta Llama 3 8B Instruct demonstrated higher overall F1 scores (averaging around 0.75–0.76), showing strong alignment with human-authored requirements, while DeepSeek 7B Chat achieved competitive but comparatively lower coverage (average F1 ≈ 0.67). Dataset-specific variations were also observed; for instance, CCHIT consistently achieved higher coverage due to its rich and well-defined HLR–LLR mappings, while GANTT performed weaker, reflecting the difficulty of handling abstract or sparse datasets.

These findings confirm that LLMs have promising potential for semi-automated requirements engineering, particularly for supporting coverage analysis and accelerating requirements decomposition. However, challenges remain in handling domain-specific terminology, variability in output quality, and capturing subtle semantic nuances. Addressing these gaps will require targeted fine-tuning, refined prompting methodologies, and human-in-the-loop validation.

Overall, this chapter establishes the feasibility of using LLMs for HLR-to-LLR coverage analysis, combining automated semantic similarity metrics with qualitative observations. The inclusion of structured evaluation tables strengthens the evidence base, while the comparative analysis across datasets highlights both the opportunities and limitations of current LLMs in this domain. These insights lay a strong foundation for advancing toward hybrid frameworks that integrate automated generation with expert-driven validation, as explored further in the concluding chapter of this thesis.

Chapter 6

Conclusion

The Conclusion chapter is a critical reflection on the research conducted, encapsulating the key findings, contributions, and broader implications of the work. This final chapter serves to tie together the insights gained from our study, while also addressing its limitations and proposing potential directions for future research.

6.1 Restating Research Objectives and Questions

The primary objective of this research was to explore the application of large language models (LLMs) in automating the process of High-Level Requirement (HLR) to Low-Level Requirement (LLR) coverage evaluation. Specifically, the research aimed to investigate the feasibility of using LLMs, such as Gemini 2.0 Flash [17], Meta Llama 3 8B Instruct [1] and DeepSeek 7B Chat **deepseek7Bchat**, to generate LLRs from given HLRs and assess the coverage between these two levels of requirements. Additionally, the research sought to evaluate the performance of LLM-generated LLRs compared to human-generated LLRs across various datasets.

The research questions guiding this study were:

- **RQ1:** How well can LLMs generate LLRs from HLRs using various prompting strategies?
- **RQ2:** What is the coverage of LLM-generated LLRs when compared to human-generated LLRs across multiple datasets?
- **RQ3:** How does the use of different prompting strategies impact the quality of LLR generation?

These research objectives and questions framed the study, guiding the methodology, and shaping the experimental design. The results presented in this thesis serve to address these questions and provide insights into the potential of LLMs in the requirements engineering process.

6.2 Summary of Key Findings

This research successfully replicated and applied multiple prompting strategies to three pre-trained models—Gemini 2.0 Flash, Meta Llama 3 8B Instruct, and DeepSeek 7B Chat—for the task of generating low-level requirements (LLRs) from high-level requirements (HLRs). The experiments explored three prompting strategies: Zero-Shot (zS), Zero-Shot with Explanation (zS + expl.) and Chain-of-Thought (CoT). The evaluation incorporated both qualitative analysis of LLR coherence and quantitative analysis using precision, recall, and F1 scores across five benchmark datasets (CM1, CCHIT, GANTT, IP, and WARC).

The findings revealed that:

- All three models demonstrated the capability to generate coherent and traceable LLRs from HLRs, with notable variations in coverage quality depending on the dataset and prompting strategy.
- The **Chain-of-Thought** and **Zero-Shot with Explanation** strategies consistently yielded more contextually accurate and logically structured LLRs, outperforming plain Zero-Shot prompts.
- **Gemini 2.0 Flash** and **Meta Llama 3 8B Instruct** achieved the strongest overall performance, with average F1 scores of approximately 0.75–0.76, while **DeepSeek 7B Chat** trailed slightly with an average F1 of around 0.67.
- Dataset-specific differences were observed: CCHIT exhibited the highest coverage and alignment (F1 $\approx$ 0.85 with Meta Llama), while GANTT performed weakest (F1 $\approx$ 0.62), reflecting the difficulty of mapping abstract or sparse HLRs.

These findings suggest that large language models (LLMs) hold strong potential in supporting the automation of LLR generation and coverage analysis. Nevertheless, differences in model performance and dataset sensitivity highlight the need for further fine-tuning, domain adaptation, and human-in-the-loop validation to ensure reliability in real-world applications.

6.3 Discussion of Implications

The results of this study have important implications for the field of requirements engineering, particularly in the context of automating requirement analysis tasks. The ability to use LLMs to generate LLRs from HLRs could significantly reduce the time and effort required for requirements engineers to manually assess requirement coverage. This automation could improve the accuracy and efficiency of requirements analysis processes, which are critical in software development.

Additionally, the findings contribute to the growing body of literature on the application of artificial intelligence (AI) in requirements engineering. By demonstrating the effectiveness of LLMs in generating and evaluating LLRs, this study provides a foundation for future research into AI-driven tools that can assist in the requirements engineering lifecycle. The results also highlight the importance of selecting appropriate prompting strategies and models for specific tasks, underscoring the need for further exploration into optimal prompting methodologies.

6.4 Contributions of the Research

This research makes several key contributions:

- It demonstrates the feasibility of using LLMs for automating the process of generating LLRs from HLRs, a task traditionally performed manually by requirements engineers.
- The study introduces a novel approach for evaluating the coverage between HLRs and LLRs, providing valuable insights into the application of AI tools in requirement analysis.
- The research explores the impact of different prompting strategies on the quality of LLRs generated by LLMs, offering a detailed analysis of their effectiveness.
- It contributes to the academic discourse on the application of AI in requirements engineering, providing a starting point for further studies in this area.

6.5 Acknowledging Limitations

Despite the promising results, this study is not without its limitations. These limitations include:

- **Scope of models:** Three LLMs in total were tested in this research, which limits the generalizability of the findings to other models or newer versions of the models.
- **Evaluation of generated LLRs:** The evaluation of LLRs generated by LLMs was not fully conducted in this study, leaving a gap in understanding how well LLMs perform in real-world settings when compared to human-generated data.
- **Dataset limitations:** The datasets used in this study (CM1, CCHIT, GANTT, IP and WARC) may not fully represent the diversity of requirements engineering problems, which could influence the applicability of the results to other domains.

Acknowledging these limitations allows for a more contextualized interpretation of the findings and highlights areas for improvement in future research.

6.6 Suggestions for Future Research

Based on the limitations and the findings of this study, several directions for future research are suggested:

- **Model comparison:** Future studies could expand the comparison to include a broader range of LLMs, including state-of-the-art models, to assess their performance in generating LLRs across different domains.
- **LLR evaluation:** The evaluation of LLRs generated by LLMs should be expanded to include expert assessments, accuracy measures, and real-world application tests to validate the quality and utility of the generated content.
- **Diverse datasets:** Future research could focus on utilizing a more diverse set of datasets that represent various industries and requirements types to enhance the generalizability of the findings.
- **Optimizing prompting strategies:** Further studies could investigate the refinement of prompting strategies to optimize LLR generation, focusing on the combination of explanation and reasoning strategies.

By exploring these areas, future research could build upon the current study and offer more refined insights into the use of AI in requirements engineering.

6.7 Final Reflections

Conducting this research has provided valuable insights into the potential of LLMs in automating requirements engineering tasks. One of the most significant learnings from this study was the realization that, while LLMs are capable of generating LLRs with reasonable accuracy, the effectiveness of the models is highly dependent on the choice of prompting strategies. This study has highlighted the complexity and potential of using AI in the requirements engineering process, and the work lays the groundwork for further investigation into more advanced models and evaluation methods.

6.8 Concluding Remarks

In conclusion, this research demonstrates the promising potential of LLMs in automating the process of generating and evaluating LLRs from HLRs. While further evaluation is required to fully validate the findings, this study contributes to the ongoing exploration of AI's role in requirements engineering. The results provide a foundation for future work aimed at optimizing LLMs for use in real-world software development processes, ultimately leading to more efficient and accurate requirements analysis and coverage assessment. The work presented here paves the way for future advancements in AI-driven tools for the software development lifecycle.

This approach ensures that the Conclusion chapter not only summarizes the research but also thoughtfully reflects on its significance and future potential, leaving the reader with a clear and compelling understanding of the work's broader impact.

Bibliography

[1] AI@Meta, "Llama 3 model card," *AI@Meta*, 2024. [Online]. Available: https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md

[2] A. Askell et al., "A general language assistant as a laboratory for alignment," *arXiv preprint arXiv:2112.00861*, 2021.

[3] J. Cleland-Huang, A. Czauderna, M. Gibiec, and J. Emenecker, "A machine learning approach for tracing regulatory codes to product specific requirements," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1*, ser. ICSE '10, Cape Town, South Africa: Association for Computing Machinery, 2010, pp. 155–164, ISBN: 9781605587196. DOI: 10.1145/1806799.1806825 [Online]. Available: https://doi.org/10.1145/1806799.1806825

[4] J. Cleland-Huang, O. Gotel, J. H. Hayes, P. Mäder, and A. Zisman, "Software traceability: Trends and future directions," *Future of Software Engineering Proceedings*, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:7224437

[5] J. Cleland-Huang, A. Zisman, and O. Gotel, *Software and Systems Traceability*. DePaule University, Oct. 2012, pp. 1–491, ISBN: 978-1-4471-2238-8. DOI: 10.1007/978-1-4471-2239-5

[6] C. Community, *Center of excellence for software & systems traceability*, [Online; accessed 27-April-2025]. [Online]. Available: http://coest.org/

[7] DeepSeek-AI et al., *Deepseek llm: Scaling open-source language models with longtermism*, 2024. arXiv: 2401.02954 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2401.02954

[8] C. Dorn et al., “Supporting quality assurance with automated process-centric quality constraints checking,” in *IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, May 2021, pp. 1298–1310. DOI: 10.1109/ICSE43902.2021.00118

[9] S. Ezzini, S. Abualhaija, C. Arora, and M. Sabetzadeh, “Ai-based question answering assistance for analyzing natural-language requirements,” in *IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, May 2023, pp. 1277–1289. DOI: 10.1109/ICSE48619.2023.00113

[10] D. Falessi, J. Roll, J. Guo, and J. Cleland-Huang, “Leveraging historical associations between requirements and source code to identify impacted classes,” *IEEE Transactions on Software Engineering*, vol. 46, pp. 420–441, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:52047122

[11] A. Fan et al., “Large language models for software engineering: Survey and open problems,” in *IEEE/ACM International Conference on Software Engineering*, May 2023, pp. 31–53. DOI: 10.1109/ICSE-FoSE59343.2023.00008

[12] O. Gotel et al., *Traceability Fundamentals*, I. Cleland-Huang, G. J., O., and A. Zisman, Eds. Software and Systems Traceability, Springer, Berlin, 2012, pp. 3–22.

[13] O. Gotel and A. C. W. Finkelstein, “An analysis of the requirements traceability problem,” *Proceedings of IEEE International Conference on Requirements Engineering*, pp. 94–101, 1994. [Online]. Available: https://api.semanticscholar.org/CorpusID:5870868

[14] J. H. Hayes, A. Dekhtyar, and S. K. Sundaram, “Advancing candidate link generation for requirements tracing: The study of methods,” *IEEE Trans. Software Eng.*, vol. 32, no. 1, pp. 4–19, 2006. DOI: 10.1109/TSE.2006.3 [Online]. Available: http://dx.doi.org/10.1109/TSE.2006.3

[15] L. W. Hayes Jane Huffman Dekhtyar Alex, “A study of methods for textual satisfaction assessment,” *10.1007/s10664-012-9198-8*, 2013. DOI: https://doi.org/10.1007/s10664-012-9198-8

[16] E. Holbrook, J. Hayes, and A. Dekhtyar, "Toward automating requirements satisfaction assessment," in *Requirements Engineering Conference, 2009*, Oct. 2009, pp. 149–158. DOI: 10.1109/RE.2009.10

[17] G. Inc., *Gemini 2.0 flash*, [Online; accessed 27-April-2025]. [Online]. Available: https://cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/2-0-flash

[18] M. Kassab, C. Neill, and P. Laplante, *State of practice in requirements engineering: Contemporary data - innovations in systems and software engineering*, Apr. 2014. [Online]. Available: https://link.springer.com/article/10.1007/s11334-014-0232-4#citeas

[19] T. Kojima, S. (Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35, Curran Associates, Inc., 2022, pp. 22 199–22 213. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/8bb0d291acd4acf06ef112099c16f326-Paper-Conference.pdf

[20] B. Larson, J. Hatcliff, and P. Chalin, "Open source patient-controlled analgesic pump requirements documentation," in *Software Engineering in Health Care (SEHC), 2013 5th International Workshop*, May 2013, pp. 28–34. DOI: 10.1109/SEHC.2013.6602474

[21] D. Luitel, S. Hassani, and M. Sabetzadeh, "Improving requirements completeness: Automated assistance through large language models," *Requirements Engineering*, vol. 29, pp. 1–23, Mar. 2024. DOI: 10.1007/s00766-024-00416-3

[22] F. A. C. Pinheiro, "Requirements traceability," in *Perspectives on Software Requirements*, J. C. S. do Prado Leite and J. H. Doorn, Eds. Boston, MA: Springer US, 2004, pp. 91–113, ISBN: 978-1-4615-0465-8. DOI: 10.1007/978-1-4615-0465-8_5 [Online]. Available: https://doi.org/10.1007/978-1-4615-0465-8_5

[23] A.-R. Preda, C. Mayr-Dorn, A. Mashkoor, and A. Egyed, "Supporting high-level to low-level requirements coverage reviewing with large language models," in *Proceedings of the 21st International Conference on Mining Software Repositories*, ser. MSR '24, Lisbon, Portugal: Association for Computing Machinery, 2024,

pp. 242–253, ISBN: 9798400705878. DOI: 10.1145/3643991.3644922 [Online]. Available: https://doi.org/10.1145/3643991.3644922

[24] A. Rodriguez, K. Dearstyne, and J. Cleland-Huang, "Prompts matter: Insights and strategies for prompt engineering in automated software traceability," Jul. 2023. DOI: 10.48550/arXiv.2308.00229

[25] Y. Shin and J. Cleland-Huang, "A comparative evaluation of two user feedback techniques for requirements trace retrieval," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, ser. SAC '12, Trento, Italy: Association for Computing Machinery, 2012, pp. 1069–1074, ISBN: 9781450308571. DOI: 10.1145/2245276.2231943 [Online]. Available: https://doi.org/10.1145/2245276.2231943

[26] B. Wei, "Requirements are all you need: From requirements to code with llms," in *2024 IEEE 32nd International Requirements Engineering Conference (RE)*, 2024, pp. 416–422. DOI: 10.1109/RE59067.2024.00049

[27] J. Wei et al., "Chain-of-thought prompting elicits reasoning in large language models," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, ser. NIPS '22, New Orleans, LA, USA: Curran Associates Inc., 2022, ISBN: 9781713871088.

[28] L. Zhao et al., "Natural language processing for requirements engineering: A systematic mapping study," *ACM Comput. Surv.*, vol. 54, no. 3, Apr. 2021, ISSN: 0360-0300. DOI: 10.1145/3444689 [Online]. Available: https://doi.org/10.1145/3444689