MASTER'S THESIS 2024

Exploring AI-Assisted Software Development at Scania: The Role of Prompt Engineering and Regulatory Compliance

Julia Bäcklund

ISSN 1650-2884
LU-CS-EX: 2024-39

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2024-39

Exploring AI-Assisted Software Development at Scania: The Role of Prompt Engineering and Regulatory Compliance

Utforska AI-assisterad mjukvaruutveckling på Scania: betydelsen av promptteknik och regelefterlevnad

Julia Bäcklund

Exploring AI-Assisted Software Development at Scania: The Role of Prompt Engineering and Regulatory Compliance

Julia Bäcklund
julia@backlund.se

June 25, 2024

Master's thesis work carried out at Scania CV AB.

Supervisors: Maria Erman, maria.erman@scania.com
Markus Borg, markus.borg@cs.lth.se

Examiner: Emma Söderberg, emma.soderberg@cs.lth.se

Abstract

Generative AI is revolutionizing industries worldwide, and no company wants to miss the innovation train. This thesis investigates the potential of AI-assisted development at Scania, focusing on enhancing code generation through prompt engineering and ensuring compliance with the EU AI Act. The study employs a methodology containing three phases: initial exploration of AI integration at Scania, testing of various prompt engineering techniques using an AI-assistant from Azure AI Studio, and an analysis of the broader implications, including regulatory compliance with the EU AI Act.

The research identifies opportunities for AI to enhance efficiency, particularly through the use of generative AI in code generation. Among various prompt engineering techniques evaluated, Few-shot, Hybrid, Chain of Thought (CoT), and Least to Most Prompting emerge as the most effective in enhancing the accuracy and utility of generated code. These techniques prove important in optimizing the performance of Azure AI Studio's AI-assistant across a series of dynamic programming problems, highlighting the potential for tailored AI implementations to meet specific organizational needs while adhering to strict security and privacy standards.

Furthermore, the study explores the implications of the EU AI Act, investigating the need for companies to align AI deployments with forthcoming regulations, particularly in high-risk applications such as autonomous vehicles—relevant for Scania's industry. The findings suggest that while the AI-assistant used for code generation falls outside the direct scope of high-risk AI systems, its implementation must still prioritize transparency, data governance, and user trust to comply with broader regulatory and ethical standards.

In conclusion, the thesis demonstrates that prompt engineering can enhance the capability of AI-assistants in software development, while also ensuring that these advancements align with the stringent requirements of the EU AI Act. Future work should continue to refine these techniques and explore their applicability in other areas of AI-assisted development.

Keywords: AI, Artificial Intelligence, LLM, Large Language Models, Prompt Engineering, Code Generation, Generative AI, AI in Software Development, EU AI Act, AI Compliance, AI-assisted Development, Azure AI Studio

Acknowledgements

I would like to thank all the interviewees, both from Scania and the PhD student, who shared their valuable insights on the various topics in this thesis. Although their contributions are anonymous, I deeply appreciate their involvement in sharing experiences and perspectives which have been instrumental in the content and depth of this research.

I am also thankful for my supervisors Markus Borg at LTH and Maria Erman at Scania for their unwavering support and insightful feedback throughout the course of this thesis. Their guidance was crucial in shaping the research direction and methodology, making this thesis project both educational and rewarding.

Contents

Chapter 1
Introduction

This chapter presents the groundwork for exploring AI-assisted development at Scania, focusing on the integration of Artificial Intelligence (AI), particularly through Large Language Models (LLMs), in enhancing software development practices. The background section in this chapter reviews the current adoption of AI in software development, emphasizing Scania's approach and the industry-wide impact. Related work examines significant research on generative AI, prompt engineering, and the EU AI Act. The purpose and research questions outline the thesis's objectives and investigative scope. Delimitations clarify the study's focus, and the structure provides an overview of the thesis's chapters.

1.1 Background

In this time of rapid technology evolution, AI, and particularly LLMs, has developed at a transformative pace. Incorporating LLM-based solutions into development practices is becoming increasingly common. This trend is highlighted in JetBrains' yearly survey, which gathers insights from developers globally, indicating that 77% of developers use ChatGPT and 46% use GitHub Copilot, both LLM-based tools, in their development practices [13]. Incorporating AI into development practices has evolved from following a trend to being perceived as a necessity for companies to stay competitive in the market. This integration of LLMs is driven by smart, adaptive, and automated solutions that have the potential to improve efficiency, accuracy, and innovation in various industries. Exploring the use of AI-assisted development is vital in order to leverage AI capabilities and has the possibility to enhance traditional development practices. The exploration is also needed to understand the limitations and potential risks with AI integration, ensuring a balanced and responsible approach to its adoption.

Scania CV AB is a leading manufacturer in the heavy-duty vehicle industry, specializing in trucks and buses for heavy transport applications. Scania was founded in 1891 and has a long-standing history of innovation and excellence in the automotive sector. The company is

renowned for its focus on sustainability, advanced technology, and commitment to quality. Today, Scania is investing in AI and new technologies as a means to stay at the forefront of development in the industry and AI-assisted development is a current topic undergoing discussion. There are numerous possibilities, but the importance of a thorough exploration of which areas have the most potential and how an effective implementation can be executed is vital in order to make knowledge-based decisions on what type of AI-assisted development to incorporate at Scania.

The topic of the benefits and limitations of implementing AI-assisted development is a popular research subject, where specific AI tools, like for example GitHub Copilot and ChatGPT, have been researched in order to determine their effectiveness [24] [25]. Although, despite the growing use of these tools, there is a gap in research on using prompt engineering to enhance the performance of generative AI in code generation. Most existing research focuses on comparing AI tools and their effectiveness in different areas, with some attention given to the general use of prompt engineering [7]. Yet, there is little detailed exploration of prompt engineering for code generation. As companies increasingly integrate AI into their workflows, understanding how to best apply generative AI technology and utilize methodologies like prompt engineering becomes key. This thesis evaluates several prompt engineering techniques to enhance code generation capabilities with generative AI. Figure 1.1 provides a schematic overview of these techniques, detailing the input, LLM, different prompting techniques, and the output process. Additionally, the thesis examines the broader implications of using AI, particularly in light of the upcoming EU AI Act.

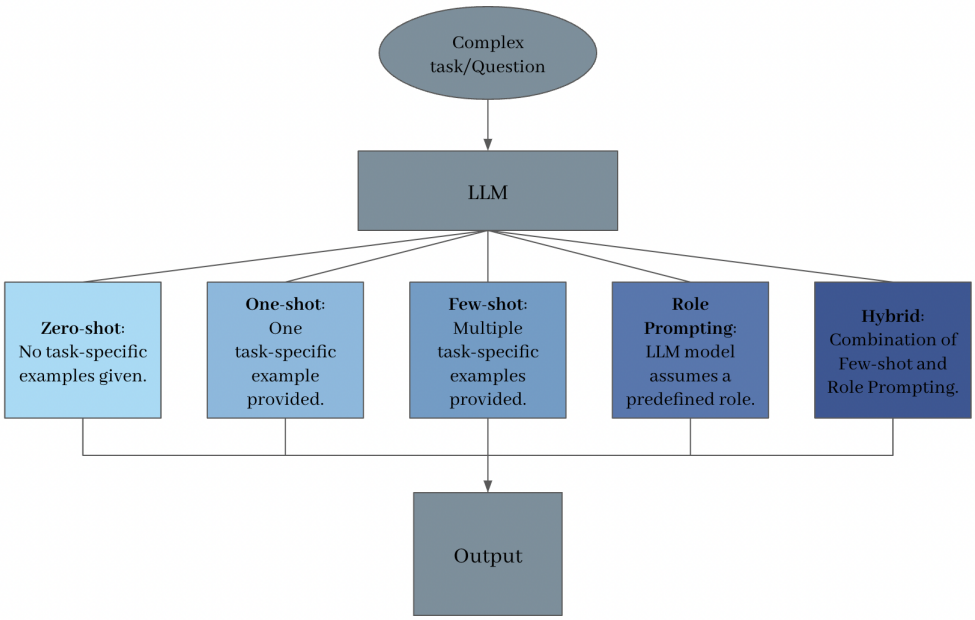


Figure 1.1: Prompt engineering processes.

1.2 Related Work

The subsections in this related work section presents relevant research for this thesis on the topics of prompt engineering, generative AI, and the EU AI Act.

1.2.1 Prompt Engineering

Prompt engineering has rapidly evolved into a crucial technique for optimizing the performance of LLMs, thereby enhancing their effectiveness across diverse applications, from programming aids to educational interfaces. The field involves designing and refining input prompts to guide LLMs in producing responses that are accurate, relevant, and contextually appropriate. Initial methods such as role-prompting and various shot techniques – zero-shot, one-shot, and few-shot – lay the groundwork by providing structured context that enhances response quality [3]. Building upon foundational techniques to prompt engineering, advanced methodologies like the Chain of Thought (CoT) and P-Tuning have further pushed the boundaries of prompt engineering. CoT prompting, for instance, guides models through logical reasoning steps, greatly improving task performance that requires deep analytical thinking [3]. Similarly, P-Tuning integrates trainable prompt embeddings to address the variability and instability typical of discrete prompts, which are fixed, unchanging text inputs, ensuring more consistent and reliable model responses.

Empirical studies such as those conducted by Denny et al. [7] explore the application of prompt engineering in educational settings, demonstrating how refined prompts can enhance the problem-solving capabilities of models like GitHub Copilot in introductory programming contexts. These studies show that prompt engineering offers practical benefits and enhances education by promoting computational thinking among learners. The work of White et al. [32] presents a Prompt Pattern Catalog which offers a comprehensive compilation of effective prompt engineering strategies that serve as reusable solutions for common challenges faced when integrating with LLMs. This catalog not only enhances the practical deployment of these models but also standardizes best practices that can be adapted across different domains, emphasizing the versatility and critical importance of prompt engineering.

Lastly, the exploration of specific innovations like P-Tuning and the detailed guides provided for effective prompt engineering with tools like ChatGPT encapsulate the specialized developments within the field [8]. These contributions highlight the depth of research and the focused efforts to refine LLM interactions, ensuring that these models not only understand and generate human-like text but do so in a way that increasingly aligns with user expectations and needs.

1.2.2 Code Generation with Generative AI

Generative AI (GenAI) technologies, particularly Large Language Models (LLMs) such as ChatGPT, have revolutionized various domains, including software engineering. The integration of these technologies facilitates tasks ranging from simple code snippet generation to complex software testing and debugging, highlighting a shift towards more automated and efficient development processes. This section explores significant contributions to the field, examining efficacy, challenges, and potential of GenAI in code generation through various scholarly works.

Initial studies, such as by Sakib et al. [25], reveal strengths in structured domains like tree-based and divide-and-conquer algorithms achieving success rates up to 71.88% in programming challenges. The same study also indicates that the debugging capabilities remain limited, with a success improvement rate of only 36.7% after feedback. This highlights a crit-

ical area for future enhancements, especially in complex problem domains such as greedy algorithms and dynamic programming where the model's performance noticeably weakens. Further research by Nejjar et al. [20] provides deeper practical insights. These studies underscore the variability in performance based on the nature of the coding tasks and the specific requirements of scientific computing and software engineering. They also discuss the importance of prompt engineering and hybrid approaches that blend traditional software engineering techniques with LLM capabilities to mitigate issues like output hallucination, which refers to the generation of logical yet incorrect or irrelevant responses by the model, and improve reliability. Innovation in prompt engineering techniques, as explored by Peng, X. [22] and Nguyen et al. [21], shows that refining prompts and integrating user feedback can significantly enhance the practical utility of LLMs in real-world applications. These studies highlight how adaptive learning strategies and user-centric design can help tailor GenAI tools to better meet the nuanced demands of software developers. This aligns with the focus of the experiment in this thesis, where similar approaches will be applied to evaluate the effectiveness and improvement of using prompt engineering for AI-assisted code generation.

Despite the promising advancements, challenges still persist, particularly in terms of the robustness and reliability of code generation techniques. The study by Mastropaolo et al. [16] illustrates how variations in natural language descriptions can lead to inconsistent outputs, emphasizing the need for clearer and more precise inputs to achieve optimal results. Additionally, studies by Yan et al. [33] and Scoccia et al. [26] delve into user experience and perceptions, highlighting mixed results in effectiveness and trustworthiness. These mixed results apply to ChatGPT's ability to consistently generate accurate and functionally reliable code across various programming challenges. The first study evaluates ChatGPT's performance across different levels of programming difficulty, finding that while the model excels in simpler, well-defined tasks, it struggles with more complex problems that require advanced logical or algorithmic reasoning. The second study gathers qualitative insights from early users who integrate ChatGPT into their coding workflows, revealing a range of experiences where some users praise its speed and utility in generating draft codes or suggestions, while others express concerns over inaccuracies and safety. These findings indicate ongoing challenges in ensuring that the code generation is not only technically correct but also practically useful and secure in diverse development environments.

As generative AI continues to evolve, so too does its application in code generation. The insights gathered from these studies not only enhance our understanding of the current landscape but also chart a course for future research endeavors. Emphasizing the need for innovation in prompt engineering, user interaction, and model training, the body of research supports a thoughtful integration of GenAI tools into software development processes. This integration should be strategic, aiming to complement rather than replace human expertise.

1.2.3 Compliance with the EU AI Act

The European Union has introduced the AI Act to establish a comprehensive regulatory framework that governs the development and deployment of artificial intelligence across its member states. This legislation aims to ensure that AI technologies are safe, transparent, and aligned with human rights and privacy standards. It categorizes AI systems based on risk, from minimal to unacceptable, requiring varying levels of compliance. High-risk applications, like those used in critical infrastructure and sensitive sectors, face the strictest

controls to prevent harm and ensure reliability and transparency. [9]

The EU AI Act requires robust compliance measures, especially for high-risk AI systems. According to the study by Walters et al. [31], these systems must undergo thorough pre-market assessments to verify their adherence to the EU's strict requirements on data governance, transparency, and technical documentation. Providers must ensure that the data feeding into AI systems is free of biases and that these systems are capable of being audited post-deployment.

The integration of the EU AI Act into the AI development and deployment represents a pivotal step towards safe, accountable, and ethically aligned AI systems. As the Act categorizes AI applications according to risk, it places specific obligations on both providers and users to ensure their AI systems operate within a secure and lawful framework. The related work discussed in this section underscores the ongoing efforts to address the challenges posed by the AI Act, particularly in the areas of software development and compliance. As developers and businesses work towards integrating these regulatory requirements, they contribute to a broader understanding of how AI can be deployed responsibly and effectively. Looking ahead, the continuous evolution of AI technologies will likely need further adaptions and refinements to these regulations to make sure that AI development is both innovative and in line with legal standards. This ongoing conversation between technology and regulation is essential for creating a space where AI can evolve while still protecting societal and ethical values.

1.3 Purpose

This thesis investigates the utilization and future opportunities of AI-assisted development at Scania, aiming to pinpoint areas where AI can boost efficiency and ensure compliance with regulations like the EU AI Act. The study progresses through phases, from identifying a suitable AI application to implementing and evaluating a proof-of-concept focused on code generation using prompt engineering.

Research Questions the thesis aims to answer the following research questions regarding the integration and impact of AI-assisted development at Scania:

- What are the potential value areas for AI-assisted development at Scania in the near term?
- How can prompt engineering be used to enhance code generation with generative AI in tasks relevant to Scania?
- What are the broader implications of integrating AI-assisted development at Scania, including compliance with the EU AI Act?

1.4 Delimitations and Scope

This thesis is centered on exploring prompt engineering for code generation and the broader implications of the EU AI Act on AI-assisted development at Scania. This research specifically evaluates the effectiveness of various prompt engineering techniques such as zero-shot,

one-shot, few-shot, role prompting, and a hybrid approach, applied to dynamic programming problems using the Azure AI Studio AI-assistant with the GPT-3.5 turbo model. This thesis aims to provide insights into prompt engineering within a specific technological setup, ensuring the relevance and applicability of the findings to Scania's operational needs.

1.5 Structure of Thesis

This thesis is structured into six main chapters, along with references and appendices.

Chapter 1: Introduction sets the stage by presenting the background, purpose, delimitations and scope, and the overall structure of the thesis.

Chapter 2: Theory delves into the critical concepts underpinning this research, including examination of ChatGPT versus and alongside Azure AI Studio, the EU AI Act, the fundamentals of Large Language Models, and the theory behind Prompt Engineering.

Chapter 3: Method outlines the methodology employed in this research, structured into three distinct phases: initial interviews and selection of the assignment, the practical application and evaluation of prompt engineering techniques with an AI-assistant adapted for code generation in a controlled setting, and a final phase that encompasses the broader implications and potential future directions.

Chapter 4: Results and Analysis presents the outcomes of the studies in the three different phases of the thesis, followed by an analysis of the results.

Chapter 5: Discussion reflects on the findings and connects them to the related work sections as well as answers the three research questions. This chapter also discusses avenues for future research and presents limitations and threats to validity for the results.

Chapter 6: Conclusion summarizes the thesis's key findings, contributions to the field, and the potential impact on AI-assisted development practices, particularly at Scania.

Chapter 2
Theory

This chapter outlines the foundational theories and methodologies related to the use of LLMs in software development, with a focus on prompt engineering and the regulatory environment by the EU AI Act. This chapter sets the stage for deeper exploration on the topics introduced.

2.1 ChatGPT or Azure AI Studio AI-assistant

Azure AI Studio is a platform within Microsoft's Azure ecosystem, designed to expedite and streamline AI development processes. It combines Azure's AI infrastructure, machine learning capabilities, cognitive services, and the OpenAI service, thereby providing a versatile environment for businesses to tailor AI solutions specific to their operational demands [17]. This architecture facilitates data upload, artifact storage, and the deployment of models which introduces a unique level of isolation and security. Each AI project operates within a distinctly isolated data container, a separation that ensures that data and resources are compartmentalized, reducing the risk of unauthorized access [18]. Azure AI Studio leverages managed virtual networks and resources controlled by Microsoft, such as computing power and data storage, to ensure its isolation features. This means it uses a secured network to protect the communication between its components and other Azure services, keeping data safe within a defined boundary.

Azure OpenAI, underpinned by Microsoft's Azure AI Services, caters primarily to enterprise needs, providing robust security features, including network isolation and private model access. One of Azure OpenAI's standout features is its adaptability to allow organizations to choose specific AI models, such as the GPT models, tailored to their unique requirements. By providing access to the GPT models Azure AI Studio enables detailed experimentation within a secure cloud and development environment. This setup enables users to leverage the full potential of GPT models while maintaining complete control over the ecosystem, something that competing services like AWS or Google Cloud, or even OpenAI's

direct offering such as ChatGPT, cannot provide.

In comparison between Azure OpenAI and ChatGPT, ChatGPT's public interface provides an accessible platform for engaging with generative AI, albeit with less emphasis on control and security measures inherent to Azure OpenAI. This distinction makes Azure OpenAI, and by extension, Azure AI Studio, more suited to enterprises looking for a secure, customizable AI development environment that can accommodate sensitive data and adhere to regulatory standards.

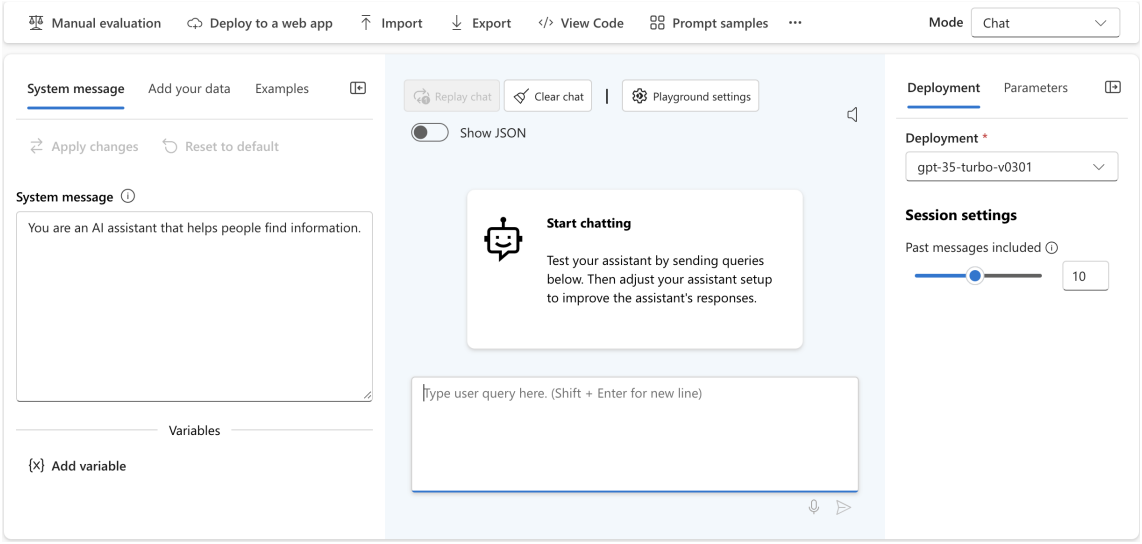


Figure 2.1: A screenshot of Azure AI Studio AI-assistant interface.

2.2 Large Language Models and the GPT-3.5 Model

LLMs are transformative tools in artificial intelligence, engineered to understand and generate human language by leveraging large amounts of textual data. Among the most prominent LLMs are the Generative Pre-trained Transformers (GPT), developed by OpenAI. These models utilize a deep learning architecture known as the transformer, which excels in capturing the complexities of language through patterns learned from extensive training datasets. Each iteration of GPT, including the GPT-3.5 model, aims to refine its predecessor's capabilities, enhancing both the depth and width of language comprehension and generation. This continuous development highlights how the model is getting better at performing tasks ranging from simple text completion to complex problem solving across various domains.

GPT-3.5, as part of the GPT series, benefits from incremental improvements over previous models, specifically tailored to handle more sophisticated language tasks. This model incorporates the latest techniques in machine learning, such as Reinforcement Learning from Human Feedback (RLHF), to fine-tune its responses based on quality feedback from human trainers. Despite these enhancements, GPT-3.5, like other LLMs, demonstrates variability in performance across different tasks. Its capabilities shine in structured settings but may struggle in more dynamic or ambiguous scenarios, such as those involving complex reasoning or

creative content generation [34]. While GPT-4 is known for being the most advanced in terms of intelligence, GPT-3.5 offers advantages in terms of speed and cost-efficiency, making it a more accessible option [27]. For this thesis, GPT-3.5 was selected after discussions with Scania, weighing the trade-offs. Its faster processing capabilities, still great performance levels, and lower cost made it the appropriate choice.

Dynamic programming represents a challenge for the GPT-3.5 model, requiring a level of problem-solving that tests the model's limits in iterative and recursive thinking [25]. Dynamic programming problems involve breaking down a problem into simpler sub-problems, solving each of these once, and storing their solution. By reusing these solutions, the model needs to efficiently solve complex problems that require combining results from previous steps, a task that is both memory and computation-intensive.

In software engineering, LLMs like GPT-3.5 can assist in coding, testing, and debugging, offering innovative approaches to traditional software tasks. However, the integration of these AI tools also brings challenges such as managing incorrect or irrelevant outputs, known as hallucinations, and ensuring that AI-generated code adheres to quality and safety standards [10].

The discussion of LLMs, particularly GPT-3.5 and its comparison with GPT-4, reveals the nuanced development of these models. While they offer unprecedented capabilities in language processing, their application in specialized fields like code generation demands continuous innovation in training methods and model handling. The exploration of these model's capabilities and limitations through research-supported analysis provides an understanding of their current state and future potential in AI-driven applications.

2.3 Prompt Engineering

Prompt engineering is the practice of strategically crafting inputs (prompts) to optimize the performance of LLMs like GPT-3.5. It plays a crucial role in achieving accurate and specific responses from models, enabling them to understand and execute tasks effectively. This technique is essential for harnessing the full potential of LLMs in a variety of applications, from customer service to complex problem-solving. The primary motivation for prompt engineering arises from the need to guide LLMs towards desired outcomes without extensive retraining. It allows for flexible adaption of models to new tasks, leveraging their generative capabilities to produce tailored outputs. This is particularly valuable in scenarios where training data is scarce or specific tasks are too niche for general models. Prompt engineering aims to enhance the efficiency and applicability of LLMs across different domains by refining their responses to fit contextual needs. However, challenges such as prompt instability – where small changes in the prompt can lead to different outcomes – make this a complex task. Addressing these issues requires a deep understanding of both the model's mechanics and the task at hand. [35] [14]

2.3.1 Prompting Techniques

In the following subsections, the theory behind the prompting techniques evaluated in this thesis is presented. Each technique described below is supported by the following sources: [32] [35] [14].

Zero-shot Learning:

Zero-shot learning involves presenting a model with a task it has not explicitly been trained to perform. For instance, while a LLM like GPT-3.5 may not be explicitly trained on dynamic programming problems, it can still attempt to generate solutions based on its understanding of coding and problem-solving strategies. This technique tests the model's ability to generalize from its training. Effective prompt engineering can refine this process by optimizing the initial prompts, thereby enhancing the model's responses without the need for specific examples.

One-shot and Few-shot Learning:

One-shot and few-shot learning techniques demonstrate tasks to the model with one or a few examples, respectively. For instance, in one-shot learning the AI-assistant might be provided with a specific dynamic programming problem and a corresponding solution. This helps the model learn the correct format and expected output for that type of problem. In few-shot learning, it receives several similar examples, each presenting a different dynamic programming problem with its respective solution. These methods refine the model's ability to interpret and solve new problems based on these examples, enhancing its capability to handle similar tasks more effectively with minimal data.

Role Prompting:

Role prompting assigns a persona to the model, guiding it to respond from the perspective of a specified role, such as a customer support agent or a software engineer. This technique tailors the model's output to align with the expectations and knowledge base of the role, enhancing relevance and specificity in its response.

Chain of Thought (CoT) Prompting:

CoT prompting encourages the model to articulate intermediate steps when tackling a problem, enhancing its ability to handle complex reasoning tasks. By structuring prompts to include sequential steps, this technique helps models generate more logical and detailed responses.

Least to Most Prompting:

This technique involves gradually increasing the complexity of prompts, starting with simpler aspects and adding more details through subsequent prompts. It is effective for complex problem-solving where building upon foundational responses is necessary.

Prompt engineering is not just a theoretical practice but a crucial component of deploying AI in real-world scenarios. It bridges the gap between a model's general capabilities and specific application requirements, ensuring that AI tools are both effective and adaptable to diverse needs. The theory and application of prompt engineering demonstrate its significant role in

maximizing the effectiveness of LLMs. By understanding and implementing various prompting techniques, developers and researchers can better utilize AI to meet specific performance criteria.

2.4 EU AI Act

The European Union Artificial Intelligence Act (EU AI Act) sets a comprehensive framework for the regulation of AI systems, specifically focusing on the implications for AI-assisted development within the EU. This legislative framework is critical in shaping the development and deployment of AI technologies, ensuring that they adhere to safety, transparency, and ethical standards. The EU AI Act categorizes AI systems based on their potential risk levels, from unacceptable to minimal risk. This classification significantly impacts developers and providers, as it determines the regulatory compliance required [30]. High-risk AI systems, such as those used in critical infrastructure or in sensitive sectors like employment and public services, are subject to the most rigorous regulations [30]. These include mandatory pre-market assessments and adherence to robust data governance standards to prevent biases and ensure transparency and fairness. [5]

In the field of AI-assisted development, particularly when navigating the complex interplay between technological innovation and regulatory compliance, understanding the specific conditions of the EU AI Act is crucial. In the following section, selected articles from the EU AI Act that are particularly related to AI-assisted development are presented. These articles are chosen because they address the challenges and responsibilities that developers must manage when integrating AI into various applications. Each article highlights different aspects of regulatory compliance, from biometric categorization to risk management and transparency requirements. By examining these articles, developers can gain insights into how to align their projects with EU regulations, ensuring that their AI solutions are both innovative and complaint with established legal standards.

- **Article 16 (Biometric Categorization)**: This article regulates AI systems that categorize individuals based on biometric data. It is vital for developers to ensure that these systems do not maintain biases or lead to discriminatory practices. [29]

- **Article 26 (Risk-Based Approach)**: This article introduces a framework that necessitates a tiered compliance structure for AI systems based on their risk levels, emphasizing the importance of aligning AI system development with specific regulatory requirements to mitigate risks associated with AI applications. [29]

- **Article 50 and 53 (High-Risk AI Systems and Provider Obligations)**: These articles outline the criteria for defining a system as high-risk and establish the obligations for AI providers, including risk management and compliance documentation. These requirements are crucial for developers at the initial stages of AI system design and throughout the development process. [29]

- **Article 55 (Transparency and Information Provision)**: This article mandates that providers ensure transparency and provide comprehensive information about the AI system's capabilities and limitations. [29]

For AI developers, especially those involved in creating or deploying high-risk AI systems, the EU AI Act necessitates a deep integration of compliance and ethical considerations into every phase of the AI development life-cycle. This includes ensuring that AI systems are designed with capabilities for record-keeping, human oversight, and that they meet high standards of accuracy, robustness, and cybersecurity. [5] Scania is dedicated to being a responsible organization that adheres to the highest development standards for their automotive software. Discussing this thesis in light of the EU AI Act underscores Scania's commitment to integrating these regulations and ethical guidelines into its AI-assisted development strategies, ensuring that innovations not only advance technological capabilities but also align with strict legal and moral standards.

Chapter 3

Method

This chapter details the methods employed to conduct this research, structured into three different phases as outlined in Figure 3.1. Each phase addresses a specific research question, employing tailored methods to gather data and insights important for the thesis. The figure illustrates the alignment of each research question with the corresponding phase and describes the methodologies implemented, ranging from interviews and setting up AI systems to literature reviews.

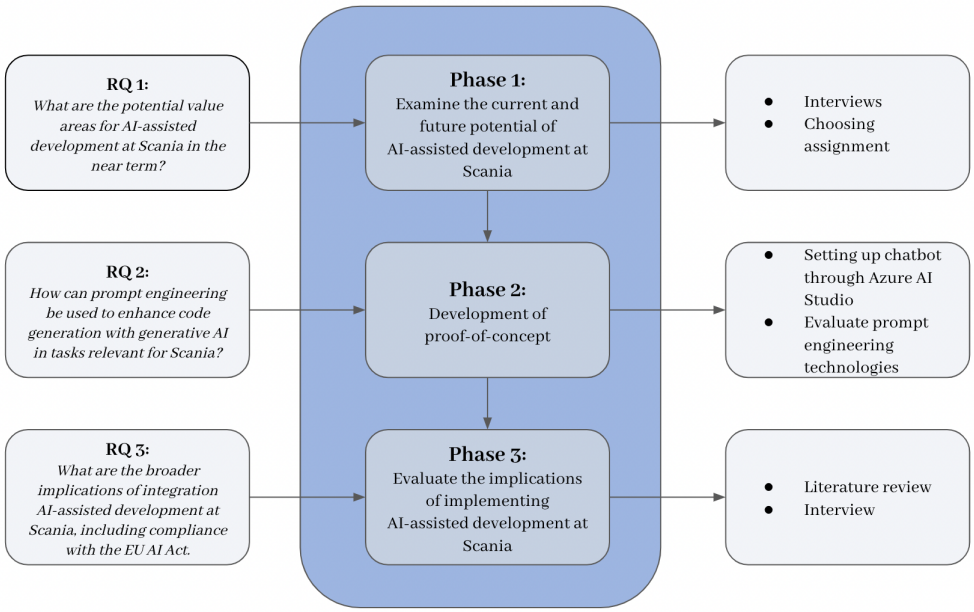


Figure 3.1: Structure of thesis project.

3.1 Phase 1

The initial phase of this thesis was dedicated to exploring the utilization and future opportunities for AI-assisted development within Scania, with a particular focus on software development practices. This exploratory phase was guided by the intent to identify potential areas where AI technologies could enhance efficiency, quality, and innovation, as well as to discover opportunities for AI-assisted development that could deliver substantial value to Scania. The exploration was constrained to areas related to software development, such as code generation, code completion, testing, and refactoring.

This phase involved consultations and interviews with Scania's IT department personnel, spanning various roles such as a scrum master, a developer, a solution architect, and a product owner. These individuals were selected based on their direct involvement or strategic insights into AI-assisted development practices. The selection process was facilitated through collaboration with the supervisor at Scania, aiming to encompass a wide array of experiences, perceptions, and anticipations regarding the use of AI in software development. The interview framework, detailed in Appendix B, was designed to extract in-depth information about the interviewees' experience with AI tools, their view on the potential of AI within their domain, and any obstacles to the adoption of external AI tools. This approach was informed by the classification from Fan et al.'s LLM review article [10], especially when discussing potential assignments for AI applications, thereby ensuring the inquiries were grounded in recognized areas of software application development.

3.1.1 Interviews

Participants the exploratory phase involved interviews with four individuals from Scania, each bringing unique perspectives and experiences related to AI-assisted development. This selection was based on purposive sampling, a method where participants are chosen because of their specific characteristics or knowledge that align with the themes of the research [2]. The participants included:

- **Scrum Master/Machine Learning Engineer** bringing 1.5 years of experience with LLMs and a total of 4 years in machine learning. The participant is pioneering the use of secure AI solutions at Scania, focusing on the Azure OpenAI API for enhanced data privacy. Their work is exploratory, aiming to set new standards in AI application and data handling within the company. Beyond professional endeavors, they recognize the significant productivity potential of generative AI, advocating for widespread familiarity with these tools among Scania employees.
- **Developer** holding 2 years of practical experience with AI/LLMs and a broader 5-year background including studies and research. The participant is working on using local open-source smaller language models to ensure data privacy and security, highlighting a commitment to innovation within Scania's security environment. They see potential in leveraging ChatGPT for solving complex coding problems and GitHub Copilot for automating mundane tasks to streamline development processes. Highlighting the importance of striking a balance between advancing technological adoption and adhering to industry-specific security standards.

- **Product Owner** relatively new to AI/LLMs but have recognized the potential and challenges of integrating external AI tools within the company, particularly around security concerns and the organization's scale. The participant advocates for the strategic use of AI in enhancing coding practices and efficiency, seeing great value in AI's ability to improve code quality and support development processes. Their insight underlines a forward-thinking approach to embedding AI technology in Scania's future, particularly emphasizing its importance in recruitment and operational efficiency.
- **Solution Architect** holding 2-3 years of experience with AI and LLMs, particularly in the context of cloud-native application architecture. The participant plays a crucial role in evaluating and implementing AI tools like GitLab Duo, GitHub Copilot, and AWS CodeWhisper at the company, with a focus on security, data privacy, and compliance with laws and regulations. This participant emphasizes the responsible use of AI to enhance development efficiency, code quality, and automation while ensuring that developers are well-informed about the technology's potential risks and benefits, underlining the importance of AI in driving Scania's future in software innovation responsibly.

Interview Structure the interviews were conducted individually via Microsoft Teams, each lasting approximately 40 minutes. The format included both open and closed questions, tailored to align with the interviewees' expertise and role-related experience with AI and LLMs.

Prepared Questions and Tailoring the questions were designed to extract insights into the current use and potential future applications of AI within Scania. The approach was flexible, allowing for adjustments based on the interviewees' backgrounds and areas of expertise. For instance, discussion with the product owner emphasized the evaluation of GitLab Duo and potential obstacles to its integration, while conversations with developers focused on areas of AI application most relevant to their work.

3.1.2 Choosing Assignment

The assignment selected for this research focuses on the evaluation of prompt engineering techniques to optimize the use of an AI-assistant for code generation tasks considered relevant for Scania. This entails setting up a custom AI-assistant environment using Azure AI Studio, ensuring isolation and enhanced security for the use of Scania's data. As previous research shows that the potential of prompt engineering, which is a low-cost technique, to substantially improve the success rate for code generation tasks, the research aims to explore the most effective methods of prompt engineering, including the evaluation of user and system prompts, as well as various prompting patterns. This exploration is intended to identify how prompt engineering can be applied to maximize the efficiency and utility of AI-assistants in software development tasks within the company. The motivation behind selecting this specific area of research emerged from a combination of insights from interviews with Scania's IT department personnel and discussions with supervisor at Scania. The dialogues underscored a recognized potential in AI-assisted development, particularly in the realm of code generation, where an interest from developers was noted alongside hesitation due to security

concerns related to external tools like GitHub Copilot and GitLab Duo. The cautious yet hopeful response to AI tools at Scania highlighted the need for a secure and efficient way to use AI in software development. This lead to the identification of a novel research gap: the application of prompt engineering within an isolated AI-assistant environment by Azure AI Studio.

Security and data privacy concerns, universally expressed across interviews, significantly shaped the decision to utilize Azure AI Studio for creating a more secure AI-assistant environment. This setup not only meets with Scania's strict data security requirements but also facilitates a more unrestricted exploration of user and system prompts using proprietary data, circumventing the restrictions present when using public platforms like ChatGPT. The evident gap in knowledge and practice concerning the optimal use of AI-assistants and prompt engineering at Scania further motivated the choice of assignment. A comprehensive, research-based approach to optimizing AI-assistant interactions had yet to be undertaken by the organization. This research is anticipated to contribute to the efficiency, security, and innovation of Scania's AI initiatives, enabling a more strategic and informed deployment of AI-assistants for development tasks.

Prompt engineering emerges as a particularly relevant area for investigation due to its potential to maximize the utility of AI-assistants already available within Scania. This relevance is amplified by the organization's current AI landscape, where the deployment of other external tools presents challenges, including the time required to integrate such technologies into a large-scale environment and security considerations. Scania highly values the ability to use AI-assistants within a controlled environment like Azure AI Studio, which facilitates secure and efficient AI-assisted development. Therefore, optimizing the use of existing in-house AI capabilities through prompt engineering presents a practical and immediate opportunity for enhancing productivity and innovation.

The foundational framework for this research will be built upon existing studies and published patterns in prompt engineering. While the assignment will not directly contrast against specific studies, it will draw inspiration from established methodologies for evaluating prompt engineering, adapting these to the unique context and technological ecosystem at Scania. This approach not only ensures the research is grounded in proven scientific principles but also tailors the exploration to meet the specific needs and challenges faced by the organization.

3.2 Phase 2

The primary objective of Phase 2 was to explore and evaluate the effectiveness of prompt engineering techniques to enhance code generation capabilities, particularly for dynamic programming problems, using Azure AI Studio's AI-assistant. This phase was pivotal in understanding how tailored prompt strategies could significantly improve the performance of generative AI models in code generation tasks. Given the need for strict security control over AI operations within Scania, the decision was made to utilize Azure AI Studio for evaluating prompt engineering techniques. Despite Scania's general use of AWS services, the specific requirements for employing powerful LLMs, particularly the GPT models, necessitated a shift from AWS to Azure. Azure AI Studio, integrated within Microsoft's ecosystem, offers exclusive access to these models with unparalleled control. This choice was influenced by

Azure's unique handling of sensitive data and its compliance with strict ISEC-regulations, essential for Scania's operational protocols. Unlike other platforms, Azure ensures that user data – including prompts, outputs, and training data – is neither accessible to other customers nor used to enhance the performance of OpenAI's or Microsoft's models [19]. This privacy guarantee, detailed by Microsoft, emphasizes that inputs and outputs are not used for model training or improving services, ensuring that Scania's data remains private and secure. Additionally, a key feature of Azure AI Studio that influenced its selection was its capability to allow modifications to the system prompt. This functionality was critical for implementing some of the prompt engineering techniques such as Role-prompting and the Hybrid approach. The ability to tailor system prompts is not universally available in all generative AI tools, making Azure AI Studio particularly suitable for this aspect of the research.

The approach for Phase 2 began with an initial evaluation of the AI-assistant's code generation capabilities, without any prompt engineering techniques applied. A series of dynamic programming problems, sourced from LeetCode, serves as the test bed for this investigation. According to studies by Sakib et al. [25], the GPT models have previously faced challenges in solving complex programming problems, particularly those categorized as Greedy and Dynamic Programming. Based on this insight, dynamic programming was selected for evaluation. This choice was driven by the need to test the AI-assistant's ability to tackle complex decision-making processes and test the LLM's ability to handle tasks that resemble real-world scenarios where decisions build progressively on earlier outcomes, which was considered relevant for Scania.

The initial phase of testing involved submitting these problems to the AI-assistant without employing any specialized prompt engineering techniques, documenting its success in resolving the problems through the generated code's performance against Leetcode's test cases. Subsequent to this baseline evaluation, various prompt engineering methods were employed, including zero-shot, one-shot, few-shot, role prompting, and a hybrid of few-shot and role prompting techniques. The effectiveness of these techniques was assessed based on the quality of the generated code, specifically its ability to pass the test cases provided by LeetCode. This metric provided a quantitative basis for assessing the quality and practical utility of the code generated by the AI-assistant under different prompting conditions. This approach allowed for a comparative analysis of the impact of each prompt engineering strategy on the AI-assistant's code generation efficiency.

Setting up AI-assistant through Azure AI Studio

The AI-assistant was set up within Azure AI Studio in collaboration with a development team at Scania, ensuring that the configuration aligns with specific requirements and standards necessary for the application. The following settings were applied during the setup process:

- Model Deployment: gpt-35-turbo
- Session History: 10 past messages included (default)
- Maximum Response Length: 800 characters (default)
- Temperature: 0.7 (default)

- Top P: 0.95 (default)
- Frequency Penalty: 0 (default)
- Presence Penalty: 0 (default)

The choice of deploying the GPT-3.5 turbo model was a decision that was made in consultation with the development team and supervisor at Scania. The model was selected due to its fast processing capabilities, good performance levels, and cost efficiency, making it a suitable choice for our needs.

3.2.1 Prompt Engineering

This section delves into the specific prompt engineering techniques used in this study. Figure 3.2 provides detailed insights into each prompting technique evaluated in this thesis. The visual representation serves as a guide to the subsequent sections on the various prompting techniques, which include zero-shot, one-shot, few-shot, role prompting, and a hybrid approach.

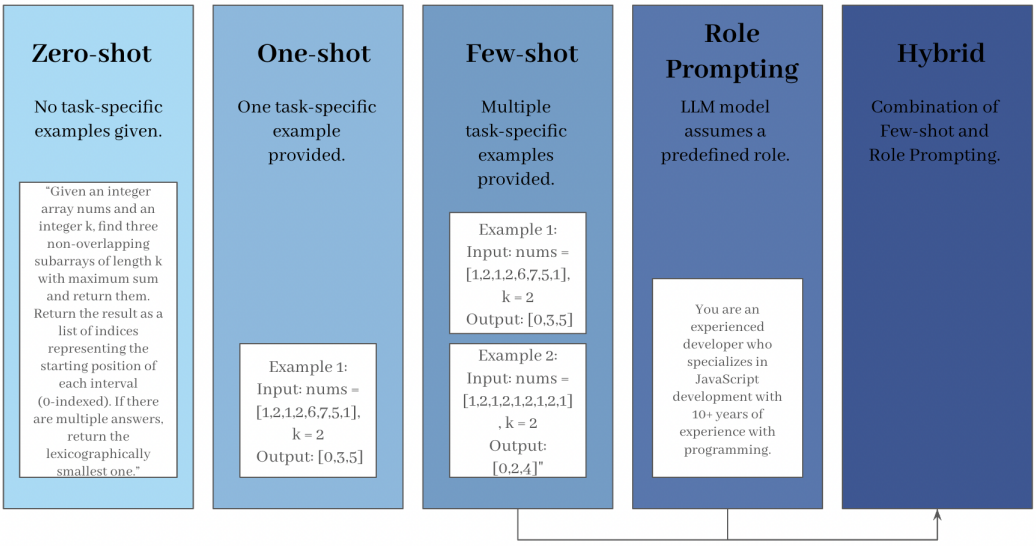


Figure 3.2: Prompting techniques.

Types of Prompt Engineering Techniques Used

Zero-shot the model is given a task without giving task-specific examples. The model generates the response based on only the prompt and the knowledge it has from previous training.

- **Example of user prompt:** *"Give me the code in JavaScript to solve this problem: Given an integer array nums and an integer k, find three non-overlapping subarrays of length k with maximum sum and return them. Return the result as a list of indices representing the starting position of each interval (0-indexed). If there are multiple answers, return the lexicographically smallest one."*

One-shot the model is given one task-specific example in the prompt. In contrast to zero-shot this approach provides an example that the model can use as a reference to structure the response.

- **Example of user prompt:** *"Give me the code in JavaScript to solve this problem: Given an integer array nums and an integer k, find three non-overlapping subarrays of length k with maximum sum and return them. Return the result as a list of indices representing the starting position of each interval (0-indexed). If there are multiple answers, return the lexicographically smallest one.*
  Example 1:
  Input: nums = [1,2,1,2,6,7,5,1], k = 2
  Output: [0,3,5]

Few-shot the model is given multiple task-specific examples in the prompt.

- **Example of user prompt:** *"Give me the code in JavaScript to solve this problem: Given an integer array nums and an integer k, find three non-overlapping subarrays of length k with maximum sum and return them. Return the result as a list of indices representing the starting position of each interval (0-indexed). If there are multiple answers, return the lexicographically smallest one.*
  Example 1:
  Input: nums = [1,2,1,2,6,7,5,1], k = 2
  Output: [0,3,5]
  Explanation: Subarrays [1, 2], [2, 6], [7, 5] correspond to the starting indices [0, 3, 5]. We could have also taken [2, 1], but an answer of [1, 3, 5] would be lexicographical larger.
  Example 2:
  Input: nums = [1,2,1,2,1,2,1,2,1], k = 2
  Output: [0,2,4]"

Role Prompting assigns a specific role to the AI by changing the system prompt, guiding its responses in a more focused manner.

- **Default system prompt:** *"You are an AI assistant that helps people find information."*
- **Example of system prompt:** *"You are an experienced developer who specializes in JavaScript development with 10+ years of experience with solving complex algorithmic problems."*

Hybrid Approach combining few-shot and role prompting leverages the strengths of both techniques.

- **Example of system prompt:** *"You are an experienced developer who specializes in JavaScript development with 10+ years of experience with solving complex algorithmic problems."*
- **Example of user prompt:** *"Give me the code in JavaScript to solve this problem: Given an integer array nums and an integer k, find three non-overlapping subarrays of length k with maximum sum and return them. Return the result as a list of indices representing the starting position of each interval (0-indexed). If there are multiple answers, return the lexicographically smallest one.*

Example 1:
Input: nums = [1,2,1,2,6,7,5,1], k = 2
Output: [0,3,5]
Explanation: Subarrays [1, 2], [2, 6], [7, 5] correspond to the starting indices [0, 3, 5]. We could have also taken [2, 1], but an answer of [1, 3, 5] would be lexicographical larger.
Example 2:
Input: nums = [1,2,1,2,1,2,1,2,1], k = 2
Output: [0,2,4]"

Programming Problems Used for Evaluation

This thesis utilized a selection of programming problems sourced from LeetCode, a online platform designed to help its users enhance their coding skills through practice and participation in coding challenges. LeetCode offers a comprehensive database of coding problems, categorized by difficulty and the type of algorithms they involve, such as dynamic programming, arrays, strings, and others. Each problems on LeetCode is accompanied by a description, a list of test cases, and a discussion board where community members can share and discuss solutions [1]. The selection of programming problems for this thesis focused, as previously mentioned, on dynamic programming problems. The problems were chosen to attempt to test the model's ability to solve problems that involve complex decision-making and optimization over sequences or arrays. The criteria for choosing these problems were twofold:

1. **Relevance to Dynamic Programming**: The initial filter applied was to select problems classified under the dynamic programming category on LeetCode. This category is extensive, containing of 499 number of problems that require breaking down a complex problem into simpler sub-problems and utilizing the solutions to these sub-problems to create a solution to the overall problem.

2. **Challenge for the LLM**: The selection of dynamic programming problems from LeetCode, categorized as easy, medium, or hard, involved a preliminary test to assess the initial difficulty for the LLM used in this thesis. The aim was to determine the AI-assistant's base capability by presenting each problem in its raw form, using only the problem description without prompt engineering techniques applied. The AI-assistant was tasked to generate code that would pass all of the provided test cases. Most of the problems that proved challenging enough to meet the criteria, and thereby necessitating the use of prompt engineering techniques, fell into the medium or hard categories. This approach was designed to ensure that the 20 selected problems would sufficiently challenge the LLM's capabilities. Note, not all available dynamic programming problems from LeetCode were tested; the selection process was stopped once 20 challenging problems were identified. This number was chosen to give a balance between providing enough data to analyze the different prompt engineering techniques and managing the workload for the evaluation within the time constraints.

The 20 dynamic programming problems that were chosen for this evaluation, together with description of the problem, an example test case, and an explanation of what the problem entails, are presented in Table 3.1 and 3.2.

Table 3.1: Dynamic programming problems used for evaluation (Part 1)

Problem ID	Description	Test Case Input	Expected Output	Explanation
1	Determine if a string is a scrambled version of another.	s1 = "great", s2 = "rgeat"	true	Given two strings of equal length, determine if one is a scrambled version of the other by recursively splitting and optionally swapping the substrings.
2	Find the minimum inserts to clear a Zuma board given 'board' and 'hand' strings.	board = "WRRBBW", hand = "RB"	-1	Determine the least number of inserts from 'hand' required to clear a Zuma board represented by 'board', by forming and eliminating trios of the same color.
3	Calculate the minimum cost to buy specific items using regular prices and discount bundles without exceeding needed quantities.	price = [2,5], special = [[3,0,5],[1,2,10]], needs = [3,2]	14	Evaluates how to smartly combine regular item prices and special bundle offers to minimize the total cost for a predefined shopping list.
4	Count decoding ways for a message with digits and '*'.	s = "*"	9	Investigates all possible interpretations of a digit-and-wildcard-encoded message, considering '*' can represent any digit from 1 to 9.
5	Find minimum operations to duplicate 'A' n times on a notepad.	n = 3	3	Calculates the optimal sequence of copy and paste actions needed to expand a single 'A' to 'n' copies with minimal steps.
6	Identify three max-sum, non-overlapping subarrays of length k.	nums = [1,2,1,2,6,7,5,1], k = 2	[0,3,5]	The task involves determining the optimal placement of three fixed-length windows on an array to maximize the aggregate sum, ensuring these windows do not overlap.

7	Count integers up to n valid and altered by 180° digit rotation.	n = 10	4	Evaluates how many numbers within a given range change to a different valid number when each digit is rotated by 180 degrees, excluding those that stay the same or become invalid.
8	Maximize score from partitioning nums into ≤k subarrays by their averages.	nums = [9,1,2,3,9], k = 3	20.00000	Focuses on dividing an array into up to k segments to maximize the combined average of these segments, accounting for every element.
9	Shortest instruction sequence to reach target from 0 with acceleration and reverse.	target = 3	2	Determines the minimal set of acceleration and reverse instructions required to navigate a car from start to a specified target on an infinite line.
10	Sum of unique characters in all substrings of s.	s = "ABC"	10	Calculates the cumulative count of unique characters across all possible substrings of a given string, including repetitions.

Table 3.2: Dynamic programming problems used for evaluation (Part 2)

Problem ID	Description	Test Case Input	Expected Output	Explanation
11	Shortest path length to visit all nodes in an undirected graph.	graph = [[1,2,3],[0],[0],[0]]	4	Identifies the minimal length path that allows visiting each node in an undirected graph, permitting revisits and edge reuse.
12	Count valid permutations for 'D' and 'I' pattern in s.	s = "DID"	5	Calculates the number of ways to arrange a sequence of integers to match a pattern of increases ('I') and decreases ('D'), considering all integers within a given range.
13	Determine outcome of a graph-based game between Mouse and Cat.	graph = [[2,5],[3],[0,4,5],[1,4,5],[2,3],[0,2,3]]	0	Assesses the strategic navigation of Mouse and Cat within an undirected graph, leading to outcomes based on optimal moves towards victory or draw conditions.

14	Count playlists avoiding repeats within k songs for goal tracks from n.	n = 3, goal = 3, k = 1	6	Explores how to construct unique playlists from a set number of songs, ensuring each is played at least once and adhering to a specified separation constraint between repeats.
15	Find smallest string containing all words as substrings.	words = ["catg","ctaagt","gcta", "ttca","atgcatc"]	"gctaagttcatgcatc"	Determines the shortest possible concatenation of a given set of words where each word must appear as a substring, avoiding redundant overlaps.
16	Minimize deletions for lexicographic order in string array.	strs = ["babca","bbazb"]	3	Aims to achieve column-wise lexicographic sorting of a string array with minimal character deletions, preserving row integrity.
17	Least operators to express x equaling target under constraints.	x = 3, target = 19	5	Calculates the minimum number of basic arithmetic operations required to transform a given number into a target value, adhering to conventional operation precedence.
18	Count integers up to n with repeated digits.	n = 20	1	Quantifies numbers within a specified range that contain one or more duplicate digits, highlighting the prevalence of repetition.
19	Max sum of two non-overlapping subarrays with lengths firstLen and secondLen.	nums = [0,6,5,2,2,5,1,9,4], firstLen = 1, secondLen = 2	20	Identifies the arrangement of two fixed-size, distinct segments within an array to achieve the highest combined sum, ensuring no overlap between the segments.
20	Smallest remaining stone weight after successive smashes.	stones = [2,7,4,1,8,1]	1	Determines the minimal possible weight of the last stone after conducting a series of pairwise smashes, with stones diminishing or being eliminated based on relative weights.

Evaluation of Prompt Engineering Techniques

The evaluation of the prompt engineering techniques was conducted through a process where each user prompt, corresponding to a specific problem and technique, was tested 10 times to assess consistency and reliability. This testing involved a total of 1000 prompts, covering the 20 programming problems across the 5 different techniques. For each series of 10 repetitions of a single user prompt, the code generated by the AI-assistant was directly tested on the LeetCode platform against all the available test cases for that problem. The number of test cases for each problem ranged between two and three test cases. In Tables 3.1 and 3.2, examples of test cases used to evaluate each problem and technique are shown. This approach highlights the reliability and efficacy of each prompting technique in producing correct and consistent code solutions.

Evaluation of Advanced Prompt Engineering Techniques

This part of the methodology is centered around the use of advanced prompt engineering techniques to refine the interaction with the Azure AI Studio AI-assistant, aiming to elevate its code generation and debugging performance. This choice was driven by the need to overcome limitations observed with simpler prompting methods during initial trials. The simpler prompting techniques – Zero-shot, One-shot, Few-shot, Role-prompting, Hybrid – provided foundational insights but were sometimes insufficient for complex problem-solving scenarios. To address these gaps, more advanced methodologies like Chain of Thought (CoT) and Least to Most Prompting were introduced [4]. These methods are designed to enhance the model's reasoning and output precision, particularly in tasks that require deeper analytical capabilities or involve multiple logical steps. The following approach was taken for this evaluation:

1. **Initial Assessment**: Evaluation of problems identified through initial trials, of simpler prompt engineering techniques, as needing further investigation. These problems were pinpointed based on their low success rates in passing all the test cases, across the simpler prompting techniques.

2. **Systematic Conversations**: To address these challenging problems, the methodology incorporates advanced prompt engineering methodologies such as Chain of Thought (CoT) and Least to Most Prompting:

    (a) **Chain of Thought Prompting**: This method involves guiding the LLM through a series of logical steps to arrive at a conclusion. By executing prompts that encourage step-by-step reasoning, such as "Let's think step by step", the model is encouraged to process and produce more detailed and accurate outputs [4].

    (b) **Least to Most Prompting**: Starting with a simpler aspect and progressively introducing more complexity, this method initially presents the AI-assistant with faulty or incorrect code from prior unsuccessful attempts under simpler techniques. This setup simulates simpler problem-solving scenarios, where the AI-assistant is tasked with identifying and correcting initial errors as a foundational step. Subsequent prompts gradually increase in complexity, asking the AI-assistant to enhance functionality, optimize performance, or correct errors [4].

3. **Iterative Refinement**: Each interaction with the AI-assistant is treated as an iterative process, where feedback from the previous responses is used to refine and adjust the prompts. This interaction aims for continuous improvement and fine-tuning of the AI-assistant's responses, leading towards a correct solution of the programming problem.

3.3 Phase 3

The final phase of the thesis marks the culminating stage of the research, where the broader implications of integrating AI-assisted development at Scania are evaluated. This phase was designed to synthesize insights from both literature reviews and an interview to answer the third research question: "What are the broader implications of integrating AI-assisted development at Scania, including compliance with the EU AI Act?". To address this, the phase is structured into two key activities: a comprehensive literature review and a targeted expert interview.

3.3.1 Literature Review

In this phase, the literature review will focus on the EU AI Act and its implications for AI-assisted development at Scania. The review will specifically examine the Act's impact on providers and deployers, and explore the implications of using LLMs, such as the GPT models, and AI environments like Azure AI Studio. Sources for this review will primarily include the EU AI Act itself, supplemented by relevant academic and industry-specific literature identified through consultations with a PhD student specializing in this area.

3.3.2 Interview

To enrich the understanding of the EU AI Act's implications, an interview was conducted with a PhD student, who specializes in this legislation. His research focuses on quality assurance and trustworthiness in the integration of machine learning models into software, particularly in the light of new regulatory frameworks like the EU AI Act. While the interviewee has no direct connection to Scania, his expertise provided valuable insights into how these regulations could impact AI development and deployment across various sectors. The interview was conducted in an informal setting and aimed to capture broad insights and recommendations for further reading.

Chapter 4
Results and Analysis

This chapter presents the results and analysis from the conducted research, organized into the three phases, each focusing towards addressing specific research questions as illustrated in Figure 3.1. Each section of this chapter delves into the outcomes of the respective phases, analyzing the collected data and providing insights into the effectiveness of the applied methodologies.

4.1 Phase 1

The initial phase of this research was important for understanding the current landscape of AI tool integration within Scania, particularly how these technologies are perceived and utilized by its users. Through a series of interviews with developers and project managers, this study aimed to uncover not only the existing use of AI applications but also the challenges and opportunities that Scania faces as its digital transformation journey. The insights gathered from these interviews are important for shaping the subsequent phases of this research, focusing on prompt engineering and the implementation of secure AI usage. Below are key findings from these interviews.

4.1.1 Interviews

An interesting revelation was the identification of a gap in utilizing AI to enhance development practices securely and efficiently. It was noted by the developer interviewed that some developers are tentatively employing AI tools like ChatGPT within controlled settings, indicative of a broader need for formal guidelines on secure and effective AI usage. This scenario presents a distinct opportunity for this research to investigate prompt engineering and secure AI usage protocols, potentially offering valuable insights to Scania by addressing these requirements.

> "We are not saying we shouldn't use AI, but we try to tell people to use it in the right way. That needs a different maturity I would say, to understand the challenges." - Solution Architect

Additionally, findings from the interviews highlighted a cautious yet optimistic perspective on the future role of AI in software development practices at Scania. There was clear recognition of ongoing proof-of-concept projects for tools like GitLab Duo and Microsoft Copilot, alongside an emphasis on organizational and regulatory challenges that decelerate the adoption of external AI technologies. Despite these hurdles, there was a consensus on the significant, yet untapped, potential of AI, particularly in areas of code generation and automated testing, to transform conventional software development processes.

> "I think everyone has to become familiar at some point with these tools because it's an enormous productivity boost that you can get." - Developer

Key Information Gathered The interviews revealed a dynamic landscape of AI applications and interest within Scania. Highlights include:

- Ongoing research and development efforts focusing on both custom AI-assistants and external AI tools.
- A recognized need for broader knowledge and understanding of AI and machine learning across the organization.
- The challenges of integrating external AI tools into Scania's operations, influenced by industry specifics and organizational size.
- The use of ChatGPT by developers, signaling interest in the efficient and secure deployment of AI-assistant technologies.
- Different perspectives on the future potential of AI in enhancing development practices, highlighting specific areas such as code generation, testing, debugging, and refactoring as key opportunities for AI to deliver substantial value to Scania.

These discussions underscored the importance of aligning AI tool adoption with Scania's strategic goals, considering security, data privacy, and the organizational learning curve.

> "We need to have very good transparency on what models are being used and what is being done with the data." - Product Owner

Challenges and Unique Insights The interviews offered unexpected insights that influenced the thesis's direction, such as the extent of personal ChatGPT usage among participants and the ongoing evaluations of various AI tools. These findings highlighted the complexity of adopting external AI technologies in large, traditional companies transitioning towards more digital and IT-centric operations. Additionally, the recognition of a knowledge gap and the need for widespread AI awareness within the company pointed towards potential areas for further research and development.

4.2 Phase 2

The second phase of this thesis presents an evaluation of various prompt engineering techniques to enhance code generation using an AI-assistant. This phase focuses on testing and comparing the effectiveness of five different simpler prompting techniques: Zero-shot, One-shot, Role prompting, Few-shot, and a Hybrid approach. Each technique was applied to a set of 20 dynamic programming problems. This phase also explores more advanced prompt engineering techniques through an iterative process tailored to address the problems that remained unsolved after applying the simpler techniques. These advanced techniques include Chain of Though Prompting and Least to Most Prompting.

4.2.1 Code Generation with AI-assistant

The Table 4.1 presents a comparison of five simpler prompt engineering techniques—Zero-shot, One-shot, Role prompting, Few-shot, and Hybrid—evaluated across 20 dynamic programming problems. Each cell in the table displays the percentage of trials (out of ten repetitions) in which the AI-assistant generated code that successfully passed all the test cases for each problem. A percentage less than 100% typically indicates that during the 10 trails, the AI-assistant variably produced one of two types of code solutions - one correct and one incorrect. Notably, in situations where the success rate was 100% the LLM consistently generated the correct solutions across all repetitions. In contrast, for lower success rates, the LLM alternated between correct and incorrect solutions, reflecting a challenge in achieving constant code generation.

The percentages help highlight the consistency of each prompting technique in achieving correct solutions under repeated tests, thus accounting for variability in the AI-assistant's performance. Meaning of the colors in the table:

- **Green** highlights good performance where the technique succeeded in more than 75% of trials.
- **Yellow** indicates moderate performance with success rates between 50% and 75%.
- **Orange** reflects poorer outcomes, with success in 25% to 50% of cases.
- **Red** denotes a very poor performance, where success was achieved in less than 25% of trials.

A comparison between the success rates of the five different prompting techniques, illustrated using a bar-graph, can be seen in Figure 4.1. Additionally, in Appendix A the Tables A.1, A.2, A.3, A.4, A.5, present the detailed results for each technique evaluated - the number of test cases passed for each of the 10 repetitions.

Advanced Prompt Engineering techniques

As indicated in Table 4.1, certain problems resulted in very poor performance for all of the prompt engineering techniques evaluated. An analysis of the complexities of these more complex programming problems is interesting to understand why these resulted in lower success rates:

Table 4.1: Comparing prompt engineering methods.

Problem	Zero-shot	One-shot	Role prompting	Few-shot	Hybrid	Total Test Cases
1	0%	100%	100%	100%	100%	3
2	20%	10%	20%	20%	10%	3
3	50%	60%	80%	70%	50%	2
4	50%	90%	80%	80%	80%	3
5	40%	40%	50%	70%	90%	2
6	40%	50%	60%	40%	40%	2
7	60%	80%	40%	80%	70%	3
8	50%	90%	80%	90%	60%	2
9	10%	40%	50%	0%	50%	2
10	0%	0%	0%	0%	0%	3
11	50%	80%	70%	90%	100%	2
12	10%	50%	10%	50%	60%	2
13	0%	0%	10%	10%	10%	2
14	70%	90%	60%	100%	100%	3
15	0%	0%	0%	20%	0%	2
16	0%	0%	0%	0%	0%	3
17	0%	0%	10%	10%	10%	3
18	60%	100%	80%	100%	100%	3
19	0%	60%	40%	80%	60%	3
20	0%	0%	0%	0%	0%	2
Median	**15.0**	**50.0**	**45.0**	**60.0**	**55.0**	-
Mean	**25.5**	**47.0**	**42.0**	**50.5**	**49.5**	-

1. **Find the Minimum Inserts to Clear a Zuma Board (Problem 2)**

   (a) **Complexity:** This problem is comparable to a combinatorial puzzle where the player must strategically place pieces (from the hand) onto the board to eliminate consecutive groups of three or more same-colored pieces. The complexity arises from the multiple possible placements and the need to predict the cascading effects of each move.

   (b) **Challenge for AI-assistant:** This task requires the LLM to engage in forward-thinking and strategy formulation, simulating various future states of the board for each potential action. This level of strategic depth and prediction can be challenging for LLMs without enhancements for sequential reasoning.

2. **Sum of Unique Characters in All Sub-strings of a String (Problem 10)**

   (a) **Complexity:** This problem requires understanding and manipulating sub-strings to calculate unique character counts across all possible combinations. The LLM must handle nested loops of logic and data accumulation, which is computationally intensive and complex in terms of state management and memory usage.

   (b) **Challenge for AI-assistant:** Keeping track of dynamic data states (like updating character counts) across multiple sub-strings is not straightforward without ex-

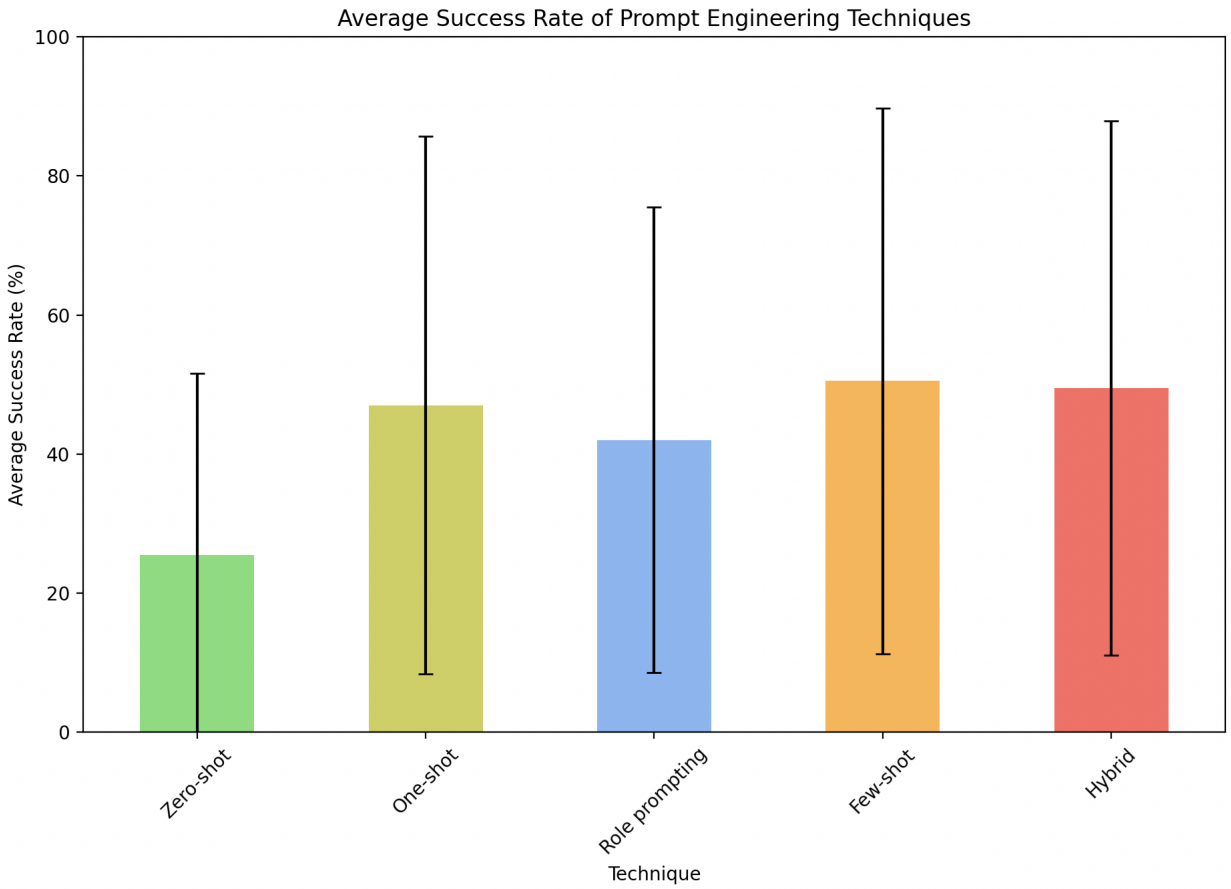


Figure 4.1: Performance of the prompt engineering techniques. The bars represent the average success rates, with error bars indicating the standard deviations.

plicit programming constructs, which can be difficult to generate correctly in a one-shot or even few-shot prompting context.

3. **Graph-Based Game Outcome Between Mouse and Cat (Problem 13)**

   (a) **Complexity:** Involves graph theory and game theory where optimal strategies must be calculated for two players with opposing goals. The solution requires not only understanding the graph structure but also predicting opponent moves, which involves recursion or backtracking.

   (b) **Challenge for AI-assistant:** Requires deep strategic thinking and potentially simulating multiple future states, tasks that exceed simple logic or loop structures and require advanced reasoning about the graph and game dynamics.

4. **Finding Smallest String Containing All Words as Sub-strings (Problem 15):**

   (a) **Complexity:** This problem involves a variation of the shortest common supersequence problem, which is NP-hard (a classification used in computational complexity theory to describe problems that are at least as hard as the hardest problems in NP, where NP stands for 'nondeterministic polynomial time'). The problem requires generating a string that contains every given word as a sub-string

in the shortest possible form, demanding complex overlap management and permutation handling.

(b) **Challenge for AI-assistant:** Involves high complexity in string manipulation, permutation, and optimization, which are challenging to encode properly through basic prompting without algorithmic guidance.

5. **Minimize Deletions for Lexicographic Order in String Array (Problem 16):**

   (a) **Complexity:** Requires dynamic programming or greedy algorithms (a class of algorithms that build up a solution piece by piece, always choosing the next piece that offers the most immediate benefit) to minimize deletions while maintaining order. It involves handling data in different forms (arrays and strings) while ensuring that the sequence of characters is maintained in alphabetical order.

   (b) **Challenge for AI-assistant:** Multi-dimensional optimization with a focus on order and minimization presents a challenge in logical structuring for an LLM without explicit algorithmic strategies.

6. **Least Operators to Express Number Equaling Target (Problem 17):**

   (a) **Complexity:** Involves finding the minimum number of operations to reach a target number from a base using only multiplication, division, addition, and subtraction. This problem can become especially complex depending on the range and operations allowed.

   (b) **Challenge for AI-assistant:** Requires numerical precision, operational order awareness, and potentially recursive calculation strategies, which are difficult for the LLM to generate accurately without deep mathematical modeling.

7. **Smallest Remaining Stone Weight After Successive Smashes (Problem 20):**

   (a) **Complexity:** This problem involves elements of both simulation and optimization, where stones must be reduced in weight through a series of decisions on which stones to smash together.

   (b) **Challenge for AI-assistant:** Simulating a series of events and adapting to ongoing changes in the data set (like the weights of stones) can be challenging. This process often requires repetitive and decision-based logic that may go beyond what standard prompting techniques can handle.

For these challenging cases, a more interactive approach was adopted to further engage with the Azure AI Studio AI-assistant in an attempt to bridge the gap between the initial unsuccessful attempts and achieving a successful code generation or debugging.

For each problem, the following information is conveyed:

- **Problem Description**: A brief recap of the problem statement.
- **Iteration Overview**: A concise description of each iterative step taken, including key adjustments to prompts and notable feedback provided to the AI-assistant.
- **Outcomes**: The outcome of each iteration, highlighting any improvements in the AI-assistant's response or successful resolution of the problem.

The prompting techniques used for this iterative process was:

- **Chain of Thought Prompting**: Step-by-step reasoning.
- **Least to Most Prompting**: Gradually increasing complexity, initially providing the AI-assistant with incorrect code from previous unsuccessful attempts (with the simpler prompt engineering techniques).

To illustrate this process, the iterations of two problems are presented in Table 4.2 and Table 4.3, one for each advanced technique tested. Finally, the overall results of this iterative process for each identified problem are presented in Table 4.4. As indicated in this table, the results for the advanced prompting techniques are presented for the method that required the fewest iterations to achieve a successful outcome, i.e., the generated code passed all of the required test cases. For each problem, both Chain of Thought (CoT) and Least to Most Prompting techniques were evaluated, but only the technique that led to a solution with the minimum iterations were documented in the result table. This approach was taken to highlight the most efficient prompting method for overcoming the specific challenges presented by each problem. Among these problems, problem 17 remained unsolved which can be due to its requirements for complex mathematical optimization and abstract reasoning, which can be challenging for LLM trained primarily on textual data.

Table 4.2: Chain of Thought Prompting - Example Problem 10

Iteration	User Prompt	AI-assistants Response
1	Explain step-by-step how to calculate the sum of unique characters for substrings of 'ABC'.	List all substrings, calculate unique characters for each, then sum these values.
2	Detail your calculation for each substring.	For 'A', 'B', 'C', each has 1 unique character. 'AB', 'BC', each has 2, and 'ABC' has 3.
3	Sum the unique characters from all substrings as you explained.	The total is 1+1+1+2+2+3 = 10.
4	Now, provide a code snippet that performs the calculations you described.	Generates the necessary code to implement the calculation steps.

Table 4.3: Least to Most Prompting - Example Problem 13

Iteration	User Prompt	AI-assistants Response
1	Here is an initial code snippet for solving the Mouse and Cat game problem, but it contains errors. Can you identify the issues and suggest corrections?	Identifies problems in the logic and suggests corrections.
2	Improve the code by adding basic checks for winning conditions.	Enhances the code by incorporating initial game-winning logic.
3	Finalize the function by implementing a recursive strategy to anticipate future moves.	Completes the function with a detailed and strategic approach, considering multiple future moves.

Table 4.4: Result of Problem Solving Iterations

Problem ID	Iterations	Success	Advanced technique
2	3	Yes	Least to Most Prompting
10	4	Yes	CoT Prompting
13	3	Yes	Least to Most Prompting
15	3	Yes	Least to Most Prompting
16	3	Yes	Least to Most Prompting
17	5	No	-
20	2	Yes	CoT Prompting

4.3 Phase 3

The final phase of this thesis delves into the relationship between artificial intelligence applications and the regulatory landscape shaped by the European Union's AI Act. This phase is important for understanding how the Act's strict regulations influence Scania's deployments of AI technologies like Azure AI Studio and particularly in roles that differentiate between AI providers and deployers.

4.3.1 Providers vs Deployers

In the evolving landscape of artificial intelligence regulation, the EU AI Act plays a pivotal role by establishing clear rules and responsibilities for different stakeholders. Distinguishing between the roles of providers and deployers is crucial because each group interacts with AI technologies at different stages of their life-cycle and thus has distinct capabilities to influence AI system behaviour and outcomes. In the context of the EU AI Act, the implications for AI systems providers and deployers are differentiated to ensure that each entity along the

AI value chain fulfills specific responsibilities that align with their operational roles.

For Providers:

Providers are primarily developers, creators, or manufacturers of AI systems. Under the EU AI Act, they bear the most comprehensive obligations, especially when it comes to ensuring the compliance of their AI systems before they are placed on the market. Providers are for example responsible for:

1. Conducting thorough risk assessments and ensuring their systems meet the required safety, transparency, and accountability standards as stipulated by the regulation, according to Article 9 of the EU AI Act [29].

2. Creating detailed documentation that covers the entire life-cycle of the AI systems, including data handling, training methodologies, and operational protocols to ensure compliance with fundamental rights and safety requirements, according to Article 10 of the EU AI Act [29].

3. Implementing and maintaining a high level of data governance and ensuring that their AI systems are free from biases that could lead to discrimination, according to Article 10(d) of the EU AI Act [29].

For Deployers:

In comparison, deployers are entities (companies, organizations, or individuals) that utilize AI systems within their operational processes. Although their obligations are less extensive than those of providers, deployers must for example ensure:

1. Proper implementation and use of AI systems in accordance with the Act's requirements, which includes ensuring the systems operate as intended without violating the regulatory standards, according to Article 29 of the EU AI Act [29].

2. Deployment of AI systems in a manner that respects the privacy and rights of individuals, which involve adhering to transparency requirements and providing necessary information to end-users about the AI system's capabilities and limitations, according to Article 30 of the EU AI Act [29].

3. Regular monitoring and reporting on the performance and impact of deployed AI systems to ensure ongoing compliance with the EU AI Act and addressing any issues related to the impact on fundamental rights or public safety, according to Article 31 of the EU AI Act [29].

4.3.2 Identifying High-Risk AI Systems

The EU AI Act introduces a legal framework that impacts the landscape of AI usage within the EU. This necessitates an understanding of how to identify high-risk AI systems which is essential for ensuring that AI applications operate safely, respect fundamental rights, and adhere to strict regulatory requirements.

Importance of Identification

Identifying high-risk AI systems is crucial because these systems often operate in sensitive areas where their failure or malfunction could pose significant risk to public safety, privacy, and fundamental human rights. The classification aids in ensuring compliance with the regulatory framework and serves to protect EU citizens from potential harms of unchecked AI technologies. By categorizing AI systems according to their risk levels, the Act aims to foster innovation while ensuring that technological advancements do not come at the expense of ethical standards and public trust. [11]

Knowing how to identify high-risk systems is important for any entity using AI systems. This knowledge allows entities to determine whether their systems are subject to strict regulatory requirements or not. It also informs them of the specific obligations they must meet, such as ensuring transparency, maintaining data governance, and providing adequate human oversight. Understanding whether an AI system is high-risk also influences strategic decisions about system design, deployment, and ongoing compliance measure, to ensure entities can prepare to meet legal and ethical standards.

How to Identify High-Risk Systems

To determine whether an AI system is high-risk under the EU AI Act, the following steps can be taken.

1. **Audit of AI Inventory**: Entities can begin by conducting a detailed audit of their AI tools to identify each system's functionalities and applications. This inventory helps in assessing which systems fall under the Act's scope.

2. **Risk Category Assessment**:

   (a) **Unacceptable Risk**: Operators must check if their AI systems fall under any of the categories explicitly prohibited by the Act, such as systems that use subtle, hidden techniques or those that exploit vulnerabilities of specific groups of people.

   (b) **High-Risk Evaluation**: If the AI system is not prohibited, the next step is to determine if it qualifies as high-risk. This involves assessing if the system significantly impacts safety or fundamental rights. High-risk categories include AI used in critical infrastructures, educational tools, employment and task management, law enforcement, and other sectors where risk to public safety and rights is substantial.

   (c) **General Purpose AI (GPAI)**: For systems categorized under GPAI, additional assessments are required to determine if they pose systemic risk, necessitating more stringent compliance requirements.

3. **Compliance Requirements**: If an AI system is classified as high-risk it is subject to rigorous obligations including, but not limited to, ensuring data governance, drafting technical documentation, keeping detailed records, providing transparency, ensuring human oversight, and maintaining high standards of accuracy, robustness, and cybersecurity.

4. **Lower Risk and Transparent Operations**: Systems identified with minimal risk are subject to fewer controls but are encouraged to adopt best practices through voluntary codes of conduct. Systems that interact directly with users must clearly disclose their AI-driven nature to ensure transparency.

[11]

Classification of Azure AI Studio AI-assistant

The Azure AI Studio AI-assistant, equipped with the GPT-3.5 model and utilized for prompt engineering and code generation tasks in this thesis, does not fall directly under the high-risk category as defined by the EU AI Act. This conclusion is drawn from both an expert interview with a PhD student specializing in this legislation and their scholarly article on the subject [30]. The GPT-3.5 model's versatility allows it to perform a broad range of tasks beyond code generation, qualifying as a General Purpose AI (GPAI). Its primary application for this thesis is for internal development tasks such as debugging and code generation limits its direct impact on public safety or fundamental rights, thus positioning it as a low-risk entity under the current regulatory framework. Although, as a low-risk or GPAI system, the AI-assistant still adheres to general obligations of the EU AI Act, aimed at ensuring ethical AI usage, transparency, and accountability. This includes maintaining clear communication about the AI's role within operational processes, complying with data protection laws, and ensuring the fairness and non-discrimination of outputs.

4.3.3 Impact on Autonomous Vehicles (AVs)

The EU AI Act classifies AI systems in AVs as high-risk, therefore necessitating stringent adherence to regulatory requirements. For Scania, this classification entails a thorough assessment and implementation strategy to ensure that all AI-driven functionalities in their vehicles - from navigation system to collision avoidance technologies - comply with the highest safety standards set by the EU. This can involve pre-market testing, continuous monitoring, and updates to meet safety benchmarks, reflecting the Act's focus on protecting consumers and ensuring public safety. [12]

To help manage potential risk associated with AI systems in AVs, such as unexpected system failures or security breaches, Scania can adopt a proactive risk management strategy that address both current regulatory requirements but are also adaptable to future changes in the legislative landscape. [12] The EU AI Act is likely to evolve over time, especially as technological advancements and societal expectations shift. For Scania, staying informed about potential modifications and being flexible in adapting business and development strategies accordingly is essential.

Chapter 5
Discussion

This chapter delves into the discussions surrounding the three research questions, examining the potential value areas for AI-assisted development at Scania, the application of prompt engineering to enhance code generation, and the broader implications of integrating AI-assisted development at Scania, particularly in the light of the EU AI Act. Each section aims to synthesize the findings from the data gathered during the research, analyze the results, and connect these to the related work discussed in Chapter 1.2. Future research directions and potential threats to validity of the study's findings are also discussed.

5.1 RQ1: What are the potential value areas for AI-assisted development at Scania in the near term?

The main goal of this discussion section is to evaluate how AI-assisted development could bring value to Scania in the near term, based on the findings from the initial phase.

The initial phase of this thesis was dedicated to exploring the current utilization and future potential of AI-assisted development at Scania, particularly focusing on software development practices. This phase was important for understanding how AI technologies are currently being integrated within Scania and identifying the key areas where these technologies could enhance operational efficiency, quality, and innovation. The exploration of the first phase aimed to pinpoint opportunities where AI applications could deliver substantial value, especially in fields related to code generation, code completion, testing, and refactoring.

Through a series of interviews with various stakeholders from Scania's IT department, including a scrum master, a developer, a solution architect, and a product owner, a view of the perceived benefits, challenges, and expectations surrounding AI-assisted development was formed. These stakeholders provided valuable insights into the practical aspects of AI

integration, reflecting both the optimistic prospects and the cautious approach necessitated by industry-specific constraints and organizational culture.

The initial phase revealed that Scania's IT department is engaging with a range of AI tools in various stages of proof-of-concepts and pilot testing. These tools include GitLab Duo, GitHub Copilot, Microsoft 365 Copilot, AWS Code Whisperer, and to a lesser extent, ChatGPT. Each tool is being explored for its potential to enhance efficiency and address specific needs within the company.

While the full benefits of these AI tools at Scania are yet to be quantified due to the beginning stage of their integration, the anticipated advantages are clear. These AI implementations are expected to speed up development processes, enhance the accuracy of code, and free up developer time for more complex tasks. The strategic deployment of these tools is also seen as a means to keep Scania competitive and aligned with industry innovations. Although, the integration of these external AI tools has not been without its challenges. Technical and organizational hurdles, particularly around data security and compliance with European data retention policies, have slowed the integration process. For instance, concerns about GitLab Duo's data polices have necessitated careful evaluation before wider implementation. Additionally, organizational challenges such as managing expectations and communicating the progress and purpose of AI projects have emerged, highlighting the need for internal communication and education about AI capabilities and limitations.

The stakeholders interviewed at Scania, from developers to solution architects, recognize the potential of AI to transform their workspace but also emphasize the importance of cautious and informed implementation. There is a consensus that while the adoption of external AI tools presents opportunities, it also requires thorough evaluation to ensure they meet Scania's security standards. Moreover, there is an expressed need for more educational initiatives to bridge the knowledge gap on AI usage within the company. Looking forward, the stakeholders interviewed are optimistic about the role of AI at Scania, expecting it to increasingly influence various operational areas. They anticipate that future developments such as potential adjustments in GitLab's service policies and the broader integration of AI tools, will further empower Scania's workforce and enhance operational efficiencies.

This forward-looking perspective underscores a commitment to not only adopting AI technology but doing so in a manner that is secure, efficient, and aligned with the company's long-term strategic goals.

The information gathered from the interviews highlights a shared optimism about the transformative potential of AI. The following section synthesizes these insights, focusing on the most promising value areas for AI-assisted development: code generation and testing and custom AI implementations.

Code Generation and Testing

At Scania, AI's integration into code generation and testing processes has the potential to enhance efficiency. Tools like GitHub Copilot, GitLab Duo, and similar AI-assisted technologies are viewed as pivotal in automating and accelerating these tasks. According to insights from the developer interview, these tools can reduce the time developers spend on routine coding tasks, allowing them to focus on more complex, value-adding activities. The automated testing capabilities of AI, particularly through the use of LLMs, can streamline the creation of test cases for software development. For example, an LLM can automati-

cally generate test scenarios based on the specifications and expected behaviours described in development documents [28]. As highlighted in the related work section, generative AI technologies have been identified as transformative in various software engineering tasks [20], underscoring the critical role they play in both accelerating development cycles and enhancing the quality of software deliverables.

Moreover, the introduction of prompt engineering for code generation presents another layer of optimization for these AI tools. By fine-tuning how developers interact with AI-assistants, prompt engineering can maximize the accuracy and relevance of generated code. The exploration of prompt engineering techniques aligns with Scania's goal to leverage existing generative AI tools more effectively, ensuring that these tools not only integrate into development workflows but also adhere to stringent security and data privacy standards.

Custom AI Implementations

Custom AI solutions, tailored specifically to meet Scania's security and privacy requirements, represent another area of development. Custom implementations are beneficial for Scania for maintaining data integrity and compliance but also for building trust in AI tools across the organization, thereby facilitating their broader acceptance and use. The proposed AI-assistant environment using Azure AI Studio exemplifies this approach by leveraging customizable features such as choice of model deployment, parameter settings, and integration with secure and private Azure endpoints. The customization here refers not to the training of the model on company-specific data, but to the flexibility of configuring and utilizing AI tools that maintain data integrity and adhere to strict compliance and security frameworks. This view of customization emphasizes the tailored application of AI technologies to fit specific operational contexts.

This approach is aligned with the findings from the related work section, which discusses the importance of adapting AI technologies to meet specific operational needs while addressing regulatory requirements, a theme explored in studies by Mastropaolo et al. [16].

Both code generation and custom AI implementations reflect the insights and concerns raised during the interviews with Scania's IT department personnel. The cautious yet optimistic attitude towards AI, coupled with a clear recognition of its potential benefits, underscores the need for a balanced approach that considers both innovative potential and necessary precautions. By focusing on these tailored solutions and educational initiatives, Scania can harness AI's capabilities to not only enhance its software development practices but also ensure these advancements align with the company's strategic objectives and compliance requirements.

5.2 RQ2: How can prompt engineering be used to enhance code generation with generative AI in tasks relevant to Scania?

The second phase of this thesis focused on examining how prompt engineering can improve code generation with generative AI models for tasks that are relevant to Scania.

Prompt engineering was identified as a key area for research because of its ability to fine-tune the outputs of AI models to meet specific needs, thus ensuring that the AI-generated code is accurate and effectively generated. The research was driven by the hypothesis that tailored prompt strategies could improve the AI model's efficiency in solving complex programming challenges. The focus on security and operational efficiency underscores how this phase aligns with Scania's broader goals. By improving how AI models and developers interact through adjusted prompts, the research aims to enhance the usefulness of AI tools for code generation tasks within a secure and managed setting.

The techniques that were examined during phase 2 included Zero-shot, One-shot, Role Prompting, Few-shot and a Hybrid method. Table 4.1 illustrates the success rates for each technique across twenty distinct programming problems, sourced from LeetCode. Overall, the Hybrid and Few-shot techniques demonstrated higher effectiveness, achieving full success (100% pass rate) in several cases, while Zero-shot generally showed the lowest success rates.

The comparative analysis of prompt engineering techniques revealed notable variations in performance:

- **Zero-shot**: Demonstrated the lowest effectiveness with a median success rate of 15% and a mean of 25.5%. This indicates that without any contextual examples, the LLM struggled to generate correct solutions.

- **One-shot**: Showed a moderate improvement over Zero-shot, with a median success rate of 40% and a mean of 43%. This inclusion of a single example seems to aid the model's understanding sufficiently to handle a broader range of problems more effectively.

- **Role-prompting**: Yielded mixed results similar to One-shot, with a median of 40% and a mean of 38.5%. The success of Role Prompting appears to be dependent on the specific context of the problem, indicating that its utility may be limited to scenarios where the role aligns closely with the task's requirements.

- **Few-shot**: This method generally offered more consistent and higher success rates, with a median of 40% and the highest mean of 46.5%. Multiple examples provided in the prompts clearly helped the AI understand and solve the problems more effectively, demonstrating its robustness across a range of problems.

- **Hybrid**: Combining elements from Few-shot and Role Prompting, the Hybrid method did not consistently outperform all other methods but showed a slightly higher median success rate of 45% and a mean of 44%. This suggests that while the Hybrid approach

can be highly effective, its performance may vary depending on how well the combined elements are executed and the specific nature of the problems.

In Figure 4.1, a comparison between the success rates of the different prompting techniques is displayed alongside their respective standard deviations, reflecting significant variability in each technique's performance across the multiple test scenarios. Given the diverse nature of the dynamic programming problems tackled, the high standard deviations across all techniques are not surprising. Each problem's characteristics and the AI-assistant's response to specific prompts could lead to wide fluctuations in success rates, thus explaining high standard deviations across all prompting techniques. These results underscore the importance of selecting the right technique based on the specific nature of the task rather than a one-size-fits-all approach.

The accumulated performance data, shown in Figure 4.1 suggest that more detailed and contextual prompting strategies like Few-shot and Hybrid tend to provide better results in code generation tasks, particularly in complex or nuanced programming scenarios. Few-shot's consistently higher mean success rate underlines the importance of providing the AI with a clear and comprehensive understanding of what is required, reducing ambiguity that might hinder accurate code generation.

Out of the 20 dynamic programming problems, 7 had very poor success rates by all of the simpler prompting techniques. These problems are complex due to their requirement for advanced logical reasoning, a deep understanding of mathematical or algorithmic principles, and the ability to manage and manipulate data states dynamically. Because these problems often require multi-step reasoning, recursive thought processes, and handling of complex data structures, simpler prompting techniques fail to guide the AI-assistant. Therefore, in addition to the five simpler prompting techniques, two more advanced prompt engineering techniques Chain of Thought (CoT) Prompting and Least to Most Prompting were investigated to enhance the AI-assistant's problem-solving capabilities in the more challenging scenarios. These techniques are seen in Table 4.4 as useful in overcoming the limitations of the more simpler prompt methods.

- **Chain of Thought Prompting** proved effective in guiding the AI-assistant through complex logical sequences, significantly improving the AI-assistant's ability to handle tasks requiring deep analytical reasoning. For instance, in Problem 10, this technique led to a successful resolution after four iterations, demonstrating its ability to incrementally improve the LLMs performance by structuring its reasoning process.

- **Least to Most Prompting** was effective in scenarios where a problem needed to be broken down into simpler, more manageable parts. This approach was beneficial in progressively refining the AI-assistant's responses by initially presenting a less complex problem and then gradually increasing the complexity. For example, in Problem 13, this method required three iterations to achieve success, showing its strength in situations requiring adjustments and refinements.

For Scania, the integration of prompt engineering techniques into AI-assisted code generation processes not only offers improvements in the reliability and accuracy of AI-generated code but also reflects the broader implications discussed in the foundational research. This investigation aligns with the insights presented by Denny et al. [7] and White et al. [32],

demonstrating that simpler techniques such as Few-shot and Hybrid enhance the AI-assistant's ability to generate precise code. These methods, through the use of multiple examples or a combination of strategies, provide the AI-assistant with a nuanced understanding of complex programming scenarios, leading to higher quality outputs. Particularly, the Few-shot technique, by offering several contextual examples, helps develop a deep understanding of the task, which is crucial in complex environments where precision is paramount. Similarly, the Hybrid approach blends Few-shot and Role Prompting to leverage the strengths of both contextual depth and role-specific guidance.

Moreover, the integration of advanced techniques such as Chain of Thought (CoT) Prompting and Least to Most Prompting extends the benefits of prompt engineering, refining the AI-assistant's problem-solving capabilities beyond simpler methods. These more advanced strategies address the limitations where simpler methods falter, particularly in handling complex problems like those involving deep analytical reasoning and data manipulation, as highlighted by Sakib et al. [25] in their discussion on the potential of GenAI in software engineering. Therefore, by integrating the findings of this research with the established theories from the literature, Scania can leverage AI more effectively and use technological innovations to produce improvements in productivity and innovation. The combined use of Few-shot, Hybrid, CoT, and Least to Most Prompting techniques improves the AI-assistant's capabilities in code generation which aligns with the strategic implementation of generative AI highlighted in the related works on the subject. This approach addresses the research question by demonstrating the effectiveness of both simpler and advanced prompt engineering techniques in enhancing AI-driven development at Scania.

5.3 RQ3: What are the broader implications of integrating AI-assisted development at Scania, including compliance with the EU AI Act?

In the final phase of this thesis, phase three, the exploration centered on understanding the broader implications of AI-assisted development at Scania, particularly within the framework of the EU AI Act. This investigation was important for understanding how the legislative landscape influences AI integration strategies, ensuring that AI deployments not only enhance technological capabilities but also adhere to stringent safety, transparency, and ethical standards mandated by the EU. As detailed in the related work section 1.2, the EU AI Act sets a comprehensive regulatory framework that categorizes AI systems based on risk levels and defines specific compliance requirements, particularly for high-risk applications [9]. For Scania, a leader in heavy-duty manufacturing, the act plays a crucial role in steering the development and deployment of AI technologies. This affects various sectors of the business, ranging from production processes to the development of autonomous vehicle technologies. Understanding these regulations is valuable to Scania to effectively harness AI's potential while navigating the complexities of compliance and risk management. The objective of this discussion is to synthesize the insights gathered from the research and interview focusing on the use of the AI-assistant from Azure AI Studio for code generation. This synthesis aims to

evaluate the strategic implications of integrating this AI-assisted development tool at Scania, in light of the regulatory framework established by the EU AI Act. The discussion will also consider the implications of the Act's classification of autonomous vehicles (AVs) as high-risk, exploring how this classification might affect Scania's broader AI strategies.

Compliance with the EU AI Act

As a deployer of AI systems, particularly the Azure AI Studio AI-assistant for code generation, Scania must adhere to some obligations under the EU AI Act that ensure safety, transparency, and responsible data governance. The Act categorizes AI applications based on their potential risk, imposing stricter compliance measures on high-risk applications, which typically include AI systems used in critical infrastructures or sensitive sectors. The AI-assistant from Azure AI Studio doesn't directly fall under these high-risk categories, but the role of deployer of this AI system still includes maintaining standards of transparency and data handling to comply with the broader principles of the Act [30]. In the use of the AI-assistant from Azure AI Studio, Scania can also leverage Azure OpenAI Services' built-in content filtering system to enhance compliance and safety measures. This system scrutinizes both input prompts and output completions for potentially harmful content across several categories, including hate speech, sexual content, violence, and self-harm. The content filtering systems can support Scania's adherence to ethical standards and also align with the EU AI Act's requirement for transparency and accountability in AI deployments.

Provider vs. Deployer Responsibilities

Under the EU AI Act, Scania's responsibilities are distinct based on its role as either a provider or deployer. As a deployer, Scania can utilize AI systems like the Azure AI Studio AI-assistant for internal code generation tasks. In this role, Scania is primarily responsible for ensuring that this AI tool is implemented and used within the company's operational framework according to regulatory standards. This includes adhering to transparency, monitoring of AI performance, and ensuring systems operate within set ethical guidelines without direct involvement in the development or initial testing phases of the AI models [15]. This contrasts with Scania's role as a provider of AI systems in high-risk applications like autonomous vehicles, where the responsibilities are more extensive due to the potential impact on public safety. This distinction is important as it necessitates different compliance requirements under the EU AI Act, where high-risk systems, such as those used in AVs, require thorough life-cycle management from development to deployment to ensure safety and compliance [9].

5.4 Future Research

This thesis explored the use of AI-assisted development at Scania, focusing particularly on prompt engineering and the broader implications of the EU AI Act on AI deployment. The utilization of AI-assisted development at Scania and adherence to regulatory compliance present a broad scope for future research. The following areas could be particularly interesting for future research:

Extended Application of Prompt Engineering

The potential of prompt engineering to improve AI-generated code has been demonstrated with dynamic programming problems using Azure AI Studio. Future studies can investigate extending these techniques to a broader range of programming tasks and other software development activities. A comparative analysis of the effect of different GPT models in prompt engineering could for example be of interest. This analysis could explore how various iterations, such as newer versions of the GPT series, impact the efficacy of prompt engineering technologies. Research could also assess the application of prompt engineering in areas beyond code generation, such as automated documentation and error analysis, to further boost productivity and innovation within Scania's software development processes.

Long-Term Studies on AI Deployment

To fully grasp the extended impacts of the potential of AI integration within Scania, it is important to conduct extended studies. These would track the long-term effect of AI-assisted development on productivity, innovation, and regulatory compliance, providing valuable insights into the long-term benefits and evolving challenges. By monitoring outcomes over extended periods, Scania could better understand how AI technologies adapt to and influence changes in software development practices, market dynamics, and regulatory landscapes. Furthermore, extended data collection could help identify trends, optimize processes, and formulate strategies to maximize the effectiveness of AI tools in fostering sustainable innovation and maintaining compliance with industry standards.

Implications of High-Risk AI Classification under the EU AI Act

Given the classification of AI systems in autonomous vehicles (AVs) as high-risk under the EU AI Act, it is critical to conduct an in-depth analysis of the regulatory implications for Scania. Future research could aim to develop comprehensive risk assessment and management strategies tailored specifically to AVs. This includes examining how upcoming modifications to the act might impact the deployment and utilization of AI within this high-stakes field.

5.5 Limitations and Threats to Validity

This research considers various aspects of validity throughout the design, execution, and analysis phases to ensure the integrity and applicability of the findings. However, like all research, this study faces certain limitations and threats to validity that may influence the interpretation and generalization of the results. The classifications and guidelines from Cruzes et al. [23] and Runeson et al. [6] were used to make the classification of the different types of validity identified for this thesis:

1. **Construct Validity** concerns whether the operational measures used truly reflect the theoretical constructs they are intended to represent. In this study:

    (a) **Operational Definitions and Measures**: The implementation of prompt engineering techniques and the evaluation of their effectiveness through repeated

trials are designed to mirror the real-world applications of these methods in enhancing code generation. While the use of LeetCode's dynamic programming problems helps standardize the testing environment, questioning whether passing these pre-defined test cases translates to a successful implementation in industrial applications highlights a potential limitation in construct validity. This measure may not fully capture the complexity and variability of real-world software development tasks, which could limit the generalizability of the results.

(b) **Interpretation of 'AI-assistant'**: The construct of an 'AI-assistant' was central in the discussions with the interviewees. Variations in understanding what constitutes an AI-assistant among the participants could influence the validity of the conclusions drawn from these interactions. Clarifying whether all parties shared the same definition and expectations of the AI-assistant's capabilities and limitations is crucial.

2. **Internal Validity** related to the cause-and-effect conclusions drawn from the study, ensuring that the results attributed to the experimental manipulations are indeed due to them and not other factors.

   (a) **Repeated Trials**: By performing ten repetitions for each user prompt, the study aims to minimize the effects of random variations in AI performance. Additionally, the iterative testing employed for problems that resisted initial solutions further exemplified an iterative approach to problem-solving. However, the inherent variability in generative AI responses could still influence the consistency of the results, potentially complicating the effects attributed to different prompting techniques.

3. **External Validity** addresses the generalizability of the findings beyond the specific conditions of the study.

   (a) **Selection of Tools and Technologies**: The exclusive use of Azure AI Studio and the GPT-3.5 model may limit the applicability of the findings to other AI platforms or newer models, such as GPT-4, which could behave differently or offer different features.

   (b) **Scope of Application**: The study employed multiple trials with different prompt engineering techniques to triangulate findings and provide a more comprehensive view of the effects of prompt engineering on code generation. This approach helped validate the results through varied application contexts within the same framework. However, the focus on code generation for dynamic programming problems, while relevant to Scania's needs, may not translate directly to other programming tasks or domains where AI-assisted development could be applied.

4. **Reliability** concerns the consistency of the study's results when replicated under similar conditions.

   (a) **Documentation and Procedures**: Detailed documentation of the experimental setup and the explicit coding of all procedures enhance the study's reliability. However, any ambiguity in the LLM's output interpretation or slight variations in the setup by future researchers could yield different results.

Chapter 6
Conclusion

The research in this thesis unfolded in three structured phases, each focusing on a different aspect of AI integration within Scania's operational and regulatory framework.

Key Findings

- **Phase 1** identified value areas for AI-assisted development at Scania as code generation and automated testing, and the development of custom AI implementations. These areas were recognized for their potential to enhance operational efficiency and innovation within Scania's software development practices.
- **Phase 2** highlighted that both simpler prompt engineering techniques, like Few-shot and a Hybrid approach, and advanced techniques such as Chain of Thought (CoT) and Least to Most Prompting, are effective in improving the accuracy and functionality of AI-generated code. These methods proved valuable in addressing complex programming challenges that standard AI outputs could not resolve.
- **Phase 3** revealed that while the Azure AI Studio AI-assistant employed does not fall directly under high-risk categories defined by the EU AI Act, AI systems used in autonomous vehicles (AVs) do. This distinction underscores the importance of aligning AI development and deployment strategies with strict regulatory standards to ensure compliance and safety.

The potential of expanding prompt engineering techniques suggests a promising direction for further research, particularly in extending these methods beyond code generation to broader software development applications. Further studies might explore integrating these techniques into various AI-assisted tools and LLMs. This integration, carried out by for example researchers and developers, could involve adapting existing prompt engineering techniques to new contexts or AI platforms, thereby enhancing the usability and efficacy of AI-assisted development tools.

The findings also emphasize the need for Scania to navigate the complexities of regulatory compliance thoughtfully, especially as AI technologies become increasingly integrated into high-risk areas like AVs. Future practices should continue to balance innovation with adherence to regulatory standards, ensuring responsible and effective use of AI technologies.

To conclude, by leveraging advanced prompt engineering techniques and aligning with regulatory requirements, Scania can enhance its AI-assisted software development capabilities and maintain a competitive edge in the industry.

References

[1] Leetcode problems. LeetCode, 2024. Accessed: 2024-05-17.

[2] Baltes, S. and Ralph, P. Sampling in software engineering research: a critical review and guidelines. https://doi.org/10.1007/s10664-021-10072-8.

[3] Chen, B., Zhang, Z., Langrené, N, and Zhu, S. Unleashing the potential of prompt engineering in large language models: a comprehensive review. https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,uid&db=edsarx&AN=edsarx.2310.14735&site=eds-live&scope=site.

[4] Chen, B., Zhang, Z., Langrené, N., and Zhu, S. Unleashing the potential of prompt engineering in large language models: a comprehensive review. arXiv:2310.14735.

[5] European Commission. Overview of the eu ai act. https://ec.europa.eu/overview-eu-ai-act, 2024. Accessed: 2024-04-17.

[6] Daniela S. Cruzes and Lotfi ben Othmane. Threats to validity in empirical software security research. https://sintef.brage.unit.no/sintef-xmlui/bitstream/handle/11250/2470488/SINTEF_ESORICS_STM_2017.pdf?sequence=1, 2021.

[7] Denny, P., Kumar, V., and Giacaman, N. Conversing with copilot: Exploring prompt engineering for solving cs1 problems using natural language. 2023. https://dl.acm.org/doi/abs/10.1145/3545945.3569823?casa_token=4_m9lBToI2MAAAAA:ApWGifnz49Q0DJ6oQmZ3G6KZZh30VYRQ0Nb8u88ARefpmYlwgJnecrdZhC_0sG5DBfDB29tewzTN92I.

[8] Ekin, S. Prompt engineering for chatgpt: a quick guide to techniques, tips, and best practices. AuthoreaPreprints.

[9] EU Artificial Intelligence Act. High-level summary of the eu artificial intelligence act. https://artificialintelligenceact.eu/high-level-summary/, 2024. Accessed: 2024-04-17.

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

[11] Osman Gazi Güçlütürk. How to identify high-risk ai systems according to the eu ai act. https://www.holisticai.com/blog/identify-high-risk-ai-systems-according-to-eu-ai-act, 2024. Accessed: 2024-05-02.

[12] Osman Gazi Güçlütürk and Bahadir Vural. Driving innovation: Navigating the eu ai act's impact on autonomous vehicles. https://www.holisticai.com/blog/driving-innovation-navigating-eu-ai-acts-impact-on-autonomous-vehicles, 2024. Accessed: 2024-05-03.

[13] Jet Brains. The state of developer ecosystem 2023. https://www.jetbrains.com/lp/devecosystem-2023/.

[14] Liu, X. et al. Gpt understands, too. doi:10.1016/j.aiopen.2023.08.012.

[15] D. Maninger, K. Narasimhan, and M. Mezini. Towards trustworthy ai software development assistance. https://search-ebscohost-com.ludwig.lub.lu.se/login.aspx?direct=true&AuthType=ip,uid&db=edsarx&AN=edsarx.2312.09126&site=eds-live&scope=site, 2023. Accessed: 21 December 2023.

[16] A. Mastropaolo et al. On the robustness of code generation techniques: An empirical study on github copilot. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 2149–2160, 2023.

[17] Microsoft. Azure ai studio. https://azure.microsoft.com/en-us/products/ai-studio.

[18] Microsoft. Azure ai studio architecture. https://learn.microsoft.com/en-us/azure/ai-studio/concepts/architecture.

[19] Microsoft. Data, privacy, and security for azure openai service. https://learn.microsoft.com/en-us/legal/cognitive-services/openai/data-privacy.

[20] Nejjar, M., Zacharias, L., Stiehle, F., and Weber, I. Llms for science: Usage for code generation and data analysis. 2023. arXivpreprintarXiv:2311.16733.

[21] N. Nguyen and S. Nadi. An empirical evaluation of github copilot's code suggestions. In *2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR)*, pages 1–5, 2022.

[22] Peng, X. Software development in the age of intelligence: embracing large language models with the right approach. frontiers of information technology electronic engineering. 2023.

[23] Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. https://link.springer.com/article/10.1007/s10664-008-9102-8, 2008. Accessed: 2024-05-18.

[24] Peng. S, et al. The impact of ai on developer productivity: Evidence from github copilot. 2023. https://search-ebscohost-com.ludwig.lub.lu.se/login.aspx?direct=true&AuthType=ip,uid&db=edsarx&AN=edsarx.2302.06590&site=eds-live&scope=site.

[25] Sakib, F.A., Khan, S.H., and Karim, A.H.M. Extending the frontier of chatgpt: Code generation and debugging. 2023. https://arxiv.org/abs/2307.08260.

[26] G. L. Scoccia. Exploring early adopters' perceptions of chatgpt as a code generation tool. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)*, pages 88–93, 2023.

[27] GumGum Tech. Comparative analysis of gpt models: Choosing the best gpt model for your use case. https://medium.com/gumgum-tech/comparative-analysis-of-gpt-models-choosing-the-best-gpt-model-for-your-use- 2023. Accessed: 2024-05-17.

[28] Workbox Technology. Ai-powered test automation: Future of software testing. https://medium.com/@workboxtech/ai-powered-test-automation-future-of-software-testing-76b072b4bbfc, 2021. Accessed: 2024-05-17.

[29] European Union. The european union artificial intelligence act. https://ec.europa.eu/EU-AI-Act, 2024. Accessed: 2024-04-17.

[30] Matthias Wagner, Markus Borg, and Per Runeson. Navigating the upcoming european union ai act. *IEEE Software*, 41(1):19–24, 2023.

[31] J. Walters et al. Complying with the eu ai act. https://search-ebscohost-com.ludwig.lub.lu.se/login.aspx?direct=true&AuthType=ip,uid&db=edsarx&AN=edsarx.2307.10458&site=eds-live&scope=site, 2023. Accessed: 29 December 2023.

[32] White, J. et al. A prompt pattern catalog to enhance prompt engineering with chatgpt. https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,uid&db=edsarx&AN=edsarx.2302.11382&site=eds-live&scope=site.

[33] D. Yan, Z. Gao, and Z. Liu. A closer look at different difficulty levels code generation abilities of chatgpt. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 1887–1898, 2023.

[34] Wayne Xin Zhao et al. A survey of large language models. https://github.com/RUCAIBox/LLMSurvey, 2024. Accessed: 2024-04-17.

[35] Zhou, Y. et al. Large language models are human-level prompt engineers. https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,uid&db=edsarx&AN=edsarx.2211.01910&site=eds-live&scope=site.

Appendices

Appendix A
Detailed Results From Phase 2 Evaluation

Table A.1: Results for Zero-shot Technique (number of test cases passed for each repetition)

ID	1	2	3	4	5	6	7	8	9	10	RES
1	2/3	2/3	2/3	2/3	2/3	2/3	2/3	2/3	2/3	2/3	0%
2	1/3	3/3	1/3	1/3	1/3	3/3	1/3	0/3	0/3	0/3	20%
3	0/2	2/2	2/2	2/2	0/2	2/2	2/2	0/2	0/2	0/2	50%
4	3/3	3/3	3/3	1/3	3/3	1/3	1/3	1/3	1/3	3/3	50%
5	1/2	2/2	1/2	2/2	1/2	1/2	2/2	1/2	1/2	2/2	40%
6	2/2	2/2	1/2	1/2	1/2	2/2	2/2	1/2	1/2	1/2	40%
7	3/3	3/3	3/3	3/3	3/3	2/3	3/3	2/3	2/3	2/3	60%
8	2/2	0/2	2/2	2/2	0/2	0/2	0/2	2/2	0/2	2/2	50%
9	2/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	10%
10	1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3	0%
11	2/2	1/2	1/2	2/2	2/2	2/2	2/2	1/2	1/2	1/2	50%
12	1/2	1/2	2/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	10%
13	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	0%
14	3/3	1/3	3/3	3/3	1/3	1/3	3/3	3/3	3/3	3/3	70%
15	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0%
16	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0%
17	0/3	0/3	0/3	0/3	0/3	2/3	0/3	0/3	0/3	0/3	0%
18	3/3	3/3	3/3	0/3	3/3	3/3	0/3	3/3	0/3	0/3	60%
19	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	2/3	0%
20	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0%

Table A.2: Results for One-shot Technique (number of test cases passed for each repetition)

ID	1	2	3	4	5	6	7	8	9	10	RES
1	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	100%
2	1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3	3/3	10%
3	0/2	2/2	0/2	2/2	0/2	2/2	2/2	2/2	2/2	0/2	60%
4	3/3	3/3	3/3	3/3	3/3	3/3	3/3	1/3	3/3	3/3	90%
5	1/2	2/2	1/2	1/2	1/2	1/2	1/2	2/2	2/2	2/2	40%
6	0/2	2/2	1/2	0/2	0/2	2/2	1/2	2/2	2/2	2/2	50%
7	0/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	1/3	80%
8	2/2	2/2	2/2	2/2	2/2	2/2	1/2	2/2	2/2	2/2	90%
9	0/2	2/2	0/2	2/2	0/2	2/2	2/2	0/2	0/2	0/2	40%
10	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0%
11	0/2	2/2	2/2	2/2	2/2	2/2	0/2	2/2	2/2	2/2	80%
12	1/2	2/2	2/2	2/2	1/2	1/2	2/2	2/2	1/2	1/2	50%
13	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	0%
14	3/3	0/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	90%
15	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	0%
16	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0%
17	1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3	0%
18	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	100%
19	2/3	3/3	3/3	2/3	2/3	3/3	3/3	3/3	2/3	3/3	60%
20	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	0%

Table A.3: Results for Role-prompting Technique (number of test cases passed for each repetition)

ID	1	2	3	4	5	6	7	8	9	10	RES
1	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	100%
2	0/3	1/3	1/3	1/3	3/3	3/3	1/3	0/3	1/3	1/3	20%
3	2/2	2/2	2/2	2/2	0/2	0/2	2/2	2/2	2/2	2/2	80%
4	3/3	3/3	3/3	3/3	3/3	3/3	3/3	2/3	1/3	3/3	80%
5	1/2	1/2	1/2	1/2	2/2	2/2	2/2	1/2	2/2	2/2	50%
6	2/2	2/2	0/2	2/2	2/2	0/2	2/2	0/2	0/2	2/2	60%
7	3/3	2/3	2/3	3/3	2/3	2/3	2/3	3/3	3/3	2/3	40%
8	0/2	2/2	2/2	2/2	2/2	2/2	0/2	2/2	2/2	2/2	80%
9	2/2	2/2	1/2	0/2	2/2	2/2	2/2	0/2	1/2	1/2	50%
10	1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3	0%
11	1/2	2/2	2/2	2/2	1/2	2/2	1/2	2/2	2/2	2/2	70%
12	0/2	0/2	0/2	0/2	0/2	0/2	2/2	0/2	0/2	0/2	10%
13	1/2	1/2	1/2	1/2	2/2	1/2	1/2	1/2	1/2	1/2	10%
14	1/3	3/3	1/3	3/3	3/3	3/3	3/3	1/3	3/3	1/3	60%
15	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0%
16	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0%
17	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	3/3	0/3	10%
18	3/3	3/3	3/3	3/3	3/3	3/3	3/3	0/3	0/3	3/3	80%
19	0/3	0/3	3/3	0/3	3/3	0/3	3/3	3/3	0/3	1/3	40%
20	1/2	1/2	1/2	0/2	1/2	1/2	1/2	1/2	1/2	1/2	0%

Table A.4: Results for Few-shot Technique (number of test cases passed for each repetition)

ID	1	2	3	4	5	6	7	8	9	10	RES
1	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	100%
2	1/3	1/3	0/3	1/3	2/3	3/3	3/3	0/3	0/3	1/3	20%
3	2/2	2/2	0/2	0/2	2/2	0/2	2/2	2/2	2/2	2/2	70%
4	3/3	3/3	3/3	1/3	1/3	3/3	3/3	3/3	3/3	3/3	80%
5	1/2	2/2	2/2	2/2	2/2	2/2	2/2	1/2	2/2	1/2	70%
6	2/2	2/2	0/2	2/2	0/2	2/2	0/2	0/2	0/2	0/2	40%
7	3/3	3/3	3/3	2/3	3/3	3/3	3/3	1/3	3/3	3/3	80%
8	2/2	0/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	90%
9	0/2	1/2	1/2	0/2	1/2	0/2	0/2	0/2	1/2	0/2	0%
10	1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3	0%
11	2/2	2/2	0/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	90%
12	1/2	2/2	1/2	2/2	2/2	2/2	1/2	1/2	1/2	2/2	50%
13	0/2	0/2	0/2	0/2	0/2	0/2	2/2	0/2	0/2	0/2	10%
14	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	100%
15	0/2	0/2	0/2	0/2	0/2	2/2	0/2	0/2	0/2	2/2	20%
16	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0%
17	1/3	1/3	1/3	1/3	1/3	3/3	1/3	1/3	1/3	1/3	10%
18	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	100%
19	0/3	3/3	3/3	3/3	0/3	3/3	3/3	3/3	3/3	3/3	80%
20	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0%

Table A.5: Results for Hybrid Technique (number of test cases passed for each repetition)

ID	1	2	3	4	5	6	7	8	9	10	RES
1	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	100%
2	2/3	2/3	2/3	2/3	3/3	2/3	2/3	2/3	2/3	2/3	10%
3	0/2	0/2	0/2	2/2	2/2	0/2	0/2	2/2	2/2	2/2	50%
4	1/3	3/3	3/3	1/3	3/3	3/3	3/3	3/3	3/3	3/3	80%
5	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	1/2	90%
6	0/2	2/2	2/2	0/2	0/2	0/2	2/2	0/2	0/2	2/2	40%
7	2/3	3/3	2/3	3/3	3/3	3/3	3/3	3/3	2/3	3/3	70%
8	0/2	0/2	2/2	2/2	2/2	0/2	2/2	2/2	0/2	2/2	60%
9	0/2	0/2	2/2	2/2	0/2	2/2	2/2	0/2	2/2	0/2	50%
10	1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3	0%
11	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	100%
12	2/2	1/2	2/2	1/2	2/2	1/2	2/2	2/2	2/2	1/2	60%
13	1/2	2/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	10%
14	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	100%
15	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0%
16	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0%
17	0/3	0/3	0/3	0/3	0/3	0/3	0/3	3/3	0/3	0/3	10%
18	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	100%
19	3/3	3/3	3/3	3/3	3/3	0/3	3/3	0/3	0/3	0/3	60%
20	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	0%

Appendix B
Interview Framework

B.0.1 Phase 1 Description

The initial phase will consist of exploratory work and research to identify areas within Scania where AI-assisted development can offer significant value. This will be achieved through consultation and interviews with Scania developers and experts. The goal is to choose an assignment within a specific domain at Scania where an AI application could be beneficial. We will use the classification of software application areas presented in Fan et al.'s LLM review article as input to the discussion [3], e.g., code generation, code completion, refactoring, or test case generation.

B.0.2 Overall Structure of Interviews

- Interviews will be conducted in English or Swedish depending on the preference of the interviewee.
- Interviews will be conducted on site at the Scania office (Södertälje or Stockholm) or on MS Teams, depending on the preference of the interviewee.
- The interviews will be recorded if consent is provided by the interviewees. The recordings will not be shared with anyone and will only be used for the purpose of writing notes after the interviews and will then be deleted.
- Interview questions will be sent to the interviewee prior to the interview.
- Notes will be sent to the interviewee after the interview so they have the possibility to add or comment.

B.0.3 Interview

Formal Introduction

- I am writing my master thesis on the topic of exploring the current use and future potential of AI-assisted development at Scania.
- The first phase in my thesis contains a process of gathering information from Scania employees regarding this topic.
- As an individual interviewee, you are guaranteed anonymity, no company names will be published in the resulting report.
- After the interview you will get the notes where you have the chance to remove any information or clarify yourself.

Before starting: Do you consent to the interview being recorded?

Background (Warm-up)

- Graduation year? Educational background?
- When did you join Scania? What is your role at Scania today? How long in that role?
- Previous relevant roles?
- Years of experience with AI/LLMs?
- How proficient would you consider yourself when it comes to generative AI and LLMs?

Part 1

- Can you tell me about your experience with AI at Scania?
  - What AI/LLM-based tools are you using today?
  - How is the development going in this area at Scania today?
- Do you think the transport industry (Scania's industry) is facing more challenges with incorporating AI tools into their practices in comparison to other industries? Why?
- What has been your experience in terms of the learning curve associated with AI/LLM-based tools?
- Within your team/part of the organization, how do you think AI/LLM-based tools can be beneficial to use?
  - Within which area of use do you think the tool can offer significant value?
- Are there any specific integration challenges when combining AI tools with existing software systems and databases at Scania?

- How does Scania ensure data security and privacy when using AI/LLM-based tools?
- (Developer) If you could decide, which AI tools would you want Scania to provide?

Part 2

- Are there any AI/LLM-based tools in particular that you have considered incorporating within your part of the organization?
  - Is there a roadblock for this process? (Scania policies? Rules & Regulations? Time-consuming? Lack of knowledge?)
- Can you think of a specific assignment that you would want to examine whether generative AI/LLM-based tools could support?
- (Show picture below B.1) Regarding the application of LLMs for general activities/processes in software development, could these alternatives be of interest for your part of the organization?

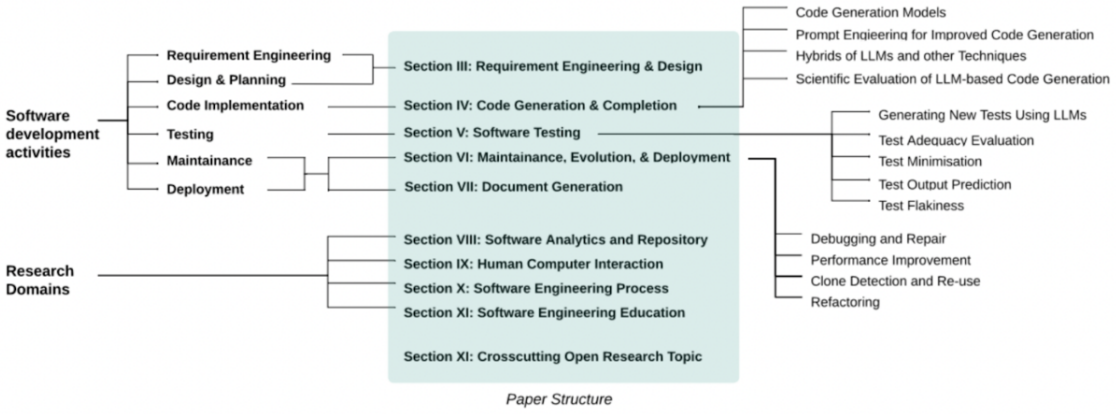


Fig. 1. A mapping between software development activities, research domains, and the paper structure

Figure B.1: Classification of Software Application Areas from Fan et al.'s LLM review article [10]

EXAMENSARBETE Exploring AI-Assisted Software Development at Scania: The Role of Prompt Engineering and Regulatory Compliance
STUDENT Julia Bäcklund
HANDLEDARE Markus Borg (LTH), Maria Erman (Scania)
EXAMINATOR Emma Söderberg (LTH)

AI i praktiken: Utforskar nya möjligheter inom mjukvaruutveckling och regelefterlevnad hos Scania

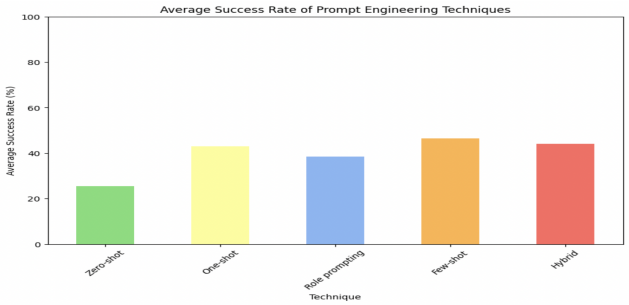
POPULÄRVETENSKAPLIG SAMMANFATTNING **Julia Bäcklund**

Genom att införa prompt engineering i AI assisterad mjukvaruutveckling har detta examensarbete vid Scania identifierat metoder för att förbättra effektiviteten och kvaliteten i genererad kod. Resultaten belyser även att den AI-assistent som använts inte klassificeras som ett högrisk AI-system enligt EU AI akten, vilket bidrar till att förstå vilka krav och föreskrifter som behöver beaktas vid användning av AI inom industrin.

Detta examensarbete har utforskat metoder för att effektivisera och förbättra kvaliteten på kodgenerering genom avancerad prompt engineering. Studien har fokuserat på att anpassa användningen av generativ AI för att möta behoven av hög produktivitet och högkvalitativ kod, centrala behov för Scanias mjukvara utvecklingsprocesser. Målet för utvärderingen var att genom att optimera hur AI används för att generera kod kunna minska tidsåtgången för utveckling samtidigt som precision och effektivitet förbättras. Ett mål som är viktigt inom en industri där kraven på regel kompatibilitet och teknisk precision är höga.

Under utvärderingen användes flera olika prompt engineering-tekniker, inklusive zero-shot, one-shot, few-shot, role prompting, hybrid teknik (en kombination av few-shot och role prompting) samt avancerade metoder som Chain of Thought och Least to Most prompting. Dessa metoder testades för att identifiera de mest effektiva sätten att förbättra AI-assistentens förmåga att generera tekniskt korrekt och användbar kod.

Av de traditionella metoderna som evaluerades



presterade few-shot och hybrid metoderna bäst i att förbättra resultatet av kodgenereringen då dessa tekniker effektivt utnyttjade fler exempel för att ge LLM:en djupare kontext och därmed mer exakt kod.

Ett oväntat inslag i studien var hur små variationer i utformningen av prompts kunde ha stor inverkan på AI-assistentens förmåga att lösa avancerade programmeringsuppgifter. Detta understryker betydelsen av att noggrant överväga hur AI-system tränas och interagerar med människor samt vilka effekter detta kan ha på den slutgiltiga produktens kvalitet.