The present work was submitted to
the Research Group
Software Construction

of the Faculty of Mathematics,
Computer Science, and
Natural Sciences

Master Thesis

LLM-based Generation of Realistic Synthetic Software Requirement Specifications

presented by

Florian Maximilian Braun

Aachen, June 1, 2025

Examiner

Prof. Dr. rer. nat. Horst Lichter

Prof. Dr. rer. nat. Bernhard Rumpe

Supervisor

Alex Sabau, M.Sc.

Acknowledgment

I would like to thank Prof. Dr. rer. nat. Horst Lichter for the opportunity to write my master thesis at his chair, as well as for his role as the first examiner. I also acknowledge Prof. Dr. rer. nat. Bernhard Rumpe for serving as the second examiner.

I am especially grateful to my supervisor, M.Sc. Alex Sabau, for his consistent support and guidance. Our regular discussions were productive and often sparked new perspectives, and I sincerely appreciated the open exchange of ideas. The collaborative environment he fostered encouraged active contribution and allowed me to gain valuable insights throughout the course of the thesis.

I would also like to thank my family for their continuous support throughout my studies and during the writing of this thesis.

Finally, I'm especially grateful to my girlfriend for her encouragement, patience, and support along the whole way.

Florian Maximilian Braun

Abstract

Access to realistic system requirement specifications (SyRS) is often limited due to confidentiality, proprietary constraints, or the high cost of expert involvement. This presents a significant challenge for software engineering tasks that depend on structured, domain-specific artefacts for evaluation, such as in the context of constraint-based recommender systems. Recent advances in large language models (LLMs) offer a potential solution for generating synthetic substitutes for real-world requirements, but systematic methods for doing so remain under-explored.

This thesis investigates how LLMs can be used to generate high-quality Synthetic System Requirement Specifications (SSRS) in domains where real-world SyRS or expert input is unavailable. To address this, a structured, repeatable process called SSRS-Gen is introduced. The process integrates scientifically grounded prompt engineering techniques, custom evaluation metrics, and LLM-based self-assessment strategies. It was applied in the context of the SecuRe security recommender system and tested across ten distinct industry domains, iteratively generating and evaluating a total of 300 SSRS instances.

Results show that prompt patterns such as Template and Persona significantly improved structural consistency and contextual plausibility of generated SSRSs. Self-assessment techniques were effective in capturing structural completeness and identifying internal inconsistencies within the SSRS, but showed limitations when evaluating realism that requires nuanced and in-depth domain-specific knowledge. Expert evaluation revealed a high degree of alignment with LLM-based assessments in many cases, while also identifying recurring weaknesses such as oversimplification, generic phrasing, and overly optimistic requirements.

This work contributes an initial, structured approach to SSRS generation using LLMs and highlights both the promise and current limitations of automated SSRS generation. It provides a foundation for future research into the role of LLMs in requirements engineering and evaluation workflows.

Contents

List of Tables

List of Figures

List of Source Codes

1. Introduction

Recent advances in large language models (LLMs) have opened new possibilities for generating complex textual artefacts across diverse industry domains. In software engineering, one particularly promising application lies in the automated creation of synthetic system requirement specifications (SSRS). These are structured documents that describe what a system should do, including its functional behaviour, external interfaces, performance parameters, and operational environment. In accordance with ISO/IEC/IEEE 29148:2018 [ISO18], a *System Requirement Specification (SyRS)* aims to define technical and non-functional requirements from a domain perspective, acting as a formal bridge between system acquirers and developers. In this thesis, SSRSs are artificially generated substitutes for such SyRS documents, intended for use in evaluation and testing scenarios where real-world specifications are unavailable and access to domain experts is limited.

The motivation for this work originates from the field of constraint-based recommender systems, particularly in the domain of software security. These systems rely on a well-defined knowledge base and require realistic input artefacts to validate performance [Fel15; Agg16]. However, acquiring real-world SyRS is often infeasible due to proprietary restrictions, confidentiality concerns, or because they are simply unavailable [FSG17]. This challenge is not unique to recommender systems but extends to many software domains where evaluation and testing depend on realistic and domain-specific requirement descriptions.

LLMs offer a compelling opportunity to address this gap. Their ability to produce coherent, domain-specific natural language outputs suggests that they may serve as substitutes for expert-authored artefacts, both in terms of content generation and evaluation. However, their successful deployment for such purposes requires more than just simple prompts. Instead, a structured process is required, informed by prompt engineering research, to guide model behaviour toward the generation of useful, high-quality SSRSs [Sch+24; Whi+23].

The process introduced in this thesis was initially developed within the context of a constraint-based software security recommender system, including the definition of a SSRS template specific to this application context. However, due to its structured, repeatable nature and the abstraction of its prompt design, the approach may be transferable to other domains that face similar challenges of data scarcity and limited expert access.

1.1. Research Questions

The central challenge addressed in this thesis concerns the generation of realistic and practically useful SSRS in domains where access to real-world SyRS or domain experts is limited. In many software engineering contexts SyRS are either inaccessible or unavailable. This lack of data poses a significant barrier for evaluation and testing processes that rely on high-quality SyRS.

Recent advancements in LLMs offer a potential solution. These models can generate structured, context-sensitive text and may be capable of producing synthetic SyRS that serve as viable substitutes for real-world documents. However, this potential raises several open questions: How can such a generation process be systematically designed? How can the quality of its outputs be meaningfully assessed, especially in the absence of expert reviewers? And most importantly, how do human experts judge the quality of artefacts that result from such a process?

To investigate these issues, the following research questions are addressed:

- **RQ 1:** How can large language models be systematically applied to generate high-quality synthetic system requirement specifications in contexts where real-world system requirement specifications are unavailable?
- **RQ 2:** How can the quality of synthetic system requirement specifications be defined and measured in the absence of human experts?
- **RQ 3:** How do human experts evaluate the quality of the LLM-generated synthetic system requirement specifications?

These questions are addressed through the development of a structured LLM-based generation process (RQ1), the formulation of automated assessment metrics (RQ2), and a follow-up expert evaluation study (RQ3).

1.2. Structure of the Thesis

The thesis is organised to build progressively from conceptual foundations to implementation and evaluation. Chapter 2 provides background on constraint-based recommender systems in general and a description of the SecuRe recommender system, which serves as the application context of this thesis. Chapter 3 introduces the foundations relevant to the thesis, covering large language models, prompt engineering techniques, and important terminology. Chapter 4 outlines the methodological approach applied in this thesis, including requirements for the process to be developed, two targeted literature reviews and the prototyping approach applied in process development. Chapter 5 presents the results of the two literature reviews as well as the rationale for which prompt engineering techniques were selected for the SSRS generation process. Chapter 6 describes the overall architecture of the generation process, including its four main phases: SSRS generation, completeness assessment, realism evaluation, and semantic similarity scoring. Chapter 7 explains how this process was implemented in practice, detailing the tools, prompt

management, and data handling used to ensure reproducibility. Chapter 8 presents the results from ten iterations of the process, analysing trends in output completeness, realism, and diversity. Chapter 9 reports on a follow-up expert study used to externally validate the realism of selected SSRSs. Chapter 10 reviews related work focusing on AI and LLMs and their application in requirement engineering contexts. Finally, Chapter 11 offers a comprehensive discussion of findings, methodological implications, and limitations, and Chapter 12 concludes the thesis by summarising key contributions and outlining directions for future research.

2. Background

To ground the research presented in this thesis, this chapter outlines the broader application context and the specific system that motivated the development of the proposed process. While the primary focus of the work is the generation of synthetic system requirement specifications (SSRSs) using large language models (LLMs), the initial idea for this process emerged from challenges encountered in evaluating constraint-based recommender systems. The chapter begins by reviewing foundational concepts in recommender system design, with particular emphasis on constraint-based approaches. Then it introduces the SecuRe recommender system and explains how the need for realistic input data within that system led to the development of SSRSs as evaluation artefacts.

Recommender systems are widely used to support decision-making across domains such as e-commerce, media consumption, and software engineering. While many recommender systems rely on user behaviour data to generate personalised suggestions, constraint-based recommender systems follow a fundamentally different approach. These systems are particularly suited to domains with complex, high-stakes decision criteria, such as software security, where user data is limited or unavailable. Instead of learning from past interactions, constraint-based systems rely on explicitly defined rules and domain knowledge to guide their recommendations. This dependency on formalised knowledge creates a significant challenge: acquiring, maintaining, and validating the underlying knowledge base often requires extensive input from domain experts [Fel15; Agg16].

In the context of such systems, there is a critical need for realistic and representative input artefacts that can be used for testing, validating, and improving the recommendation logic [Fel15]. However, access to real-world data, such as SyRS, is often restricted [FSG17]. These challenges motivated the development of a process for generating SSRS as an alternative for real-world artefacts. The following sections explain the role of constraint-based recommender systems in more detail and outline how this context motivated the development of the process proposed in this thesis.

2.1. Constraint-based Recommender Systems

Constraint-based recommender systems are a subtype of knowledge-based recommenders that operate without relying on historical user data. Instead, they derive recommendations from a formalised set of domain-specific rules and constraints encoded in a knowledge base. This makes them particularly suitable for complex and high-stakes domains, such as software security, where user interaction data is limited or unavailable and decision criteria are often highly structured and non-negotiable [Fel15; Agg16].

At the core of such systems lies the knowledge base, which comprises five key components:

- **Customer Properties** V_C: Requirements that can be elicited from a user in the given domain
- **Product Properties** V_{PROD}: Properties of products in the given domain
- **Constraints** C_R: Restrictions to the possible combinations of V_C
- **Filter Conditions** C_F: Describe possible relations between V_C and V_{PROD}
- **Product Constraints** C_{PROD}: Defines constraints to possible combinations of V_{PROD}

Recommendations are generated by solving a constraint satisfaction problem that combines the user's input (V_C) with the defined constraints to determine feasible product configurations. As such, the system's effectiveness is directly dependent on the accuracy, completeness, and consistency of the encoded knowledge [Fel15; Agg16].

However, this reliance introduces several critical challenges. First, the development of a comprehensive knowledge base typically requires substantial expert involvement, raising concerns about scalability and maintainability in large or evolving domains. Second, since most of the data is manually defined, it may reflect subjective judgments or incomplete information. These factors increase the risk of biases or inaccuracies that could propagate into the recommendation outcomes. As a result, evaluating the correctness of the knowledge base becomes essential, but also difficult, due to the lack of available ground-truth data and the complexity of the domain logic [Fel15].

Given these constraints, the next section introduces the SecuRe recommender system, which served as the initial application context for this thesis. It highlights the practical challenges of validating such systems and motivates the need for synthetic evaluation artefacts like SSRSs.

2.2. Application Context: SecuRe Recommender System

The SecuRe recommender system served as the originating context for the work presented in this thesis. It addresses critical challenges in the design of secure software systems, a domain in which security breaches can result in substantial financial and reputational damage [Mat17]. The increasing complexity of secure system architectures, combined with a significant shortage of security experts in industry, makes it difficult for software architects to make informed, domain-appropriate design decisions [FB20]. SecuRe aims to bridge this gap by recommending suitable security design artefacts based on system-specific requirements [SLL25].

As SecuRe's effectiveness relies on an accurate knowledge base, the initial motivation for this thesis was to support its evaluation through the use of synthetic, yet realistic,

system requirement specifications. Since real-world SyRSs are often inaccessible or unavailable [FSG17], the idea emerged to generate SSRSs using an LLM. These synthetic documents were intended to serve as input artefacts for validating the correctness of the SecuRe knowledge base. In doing so, they reflect the structure and terminology inherent to SecuRe and served as a concrete application context for which tailored SSRSs could be developed through the SSRS-Gen process introduced in this thesis.

2.2.1. Objective and Core Concepts

The SecuRe recommender system addresses the challenge of making expert security design knowledge more accessible, especially for software architects lacking deep expertise in security. It aims to assist in architectural decision-making by recommending suitable security patterns (SPs) based on specific security requirements and the system's contextual constraints [SLL25].

At the core of SecuRe's conceptual model lies a hierarchy of interrelated entities:

- Security Requirement:
  A specification of a condition that the system must meet to ensure a particular aspect of security.
- Security Control:
  A concrete action or technique, such as authentication, used to fulfil a security requirement and reduce system vulnerabilities.
- Security Pattern (SP):
  A reusable conceptual solution for realising a security control. SPs are abstract and implementation-independent but are characterised by a set of pattern properties such as security strength or usability.
- Security Design Pattern (SDP):
  A design-level solution of an SP.
- Realization Context:
  The set of environmental, regulatory, and architectural conditions under which the security requirement must be satisfied. It is characterised by context properties such as number of users or regulatory constraints.

SecuRe operationalises these concepts by using separate knowledge bases for each security control and security pattern. These knowledge bases encode the contextual constraints, filter conditions, and property definitions necessary to compute valid and contextually appropriate recommendations. This enables the system to solve constraint satisfaction problems and assign recommendation scores based on weighted utility criteria like cost, performance, or usability [SLL25].

This thesis focuses particularly on the concept of the realization context, which specifies the conditions under which a system must operate. These contexts serve as input

to the recommendation logic of SecuRe and influence which security patterns are considered appropriate [SLL25]. However, acquiring representative and structured data for such contexts remains difficult due to the lack of publicly available real-world data, such as SyRS [FSG17]. To address this issue, the SSRSs generated in this work are designed to simulate diverse and plausible realization contexts across different domains. These synthetic specifications are then intended to serve as input artefacts for evaluating SecuRe's knowledge base and its ability to deliver context-appropriate recommendations. To support this, a dedicated SSRS template was developed, structured as a list of context properties forming a realization context. Each of these properties is then instantiated by an LLM to generate detailed, domain-specific data.

3. Conceptual Foundations

Having established the application context and motivation in the preceding chapter, this chapter introduces the conceptual foundation necessary for understanding the remainder of the thesis. It begins with an overview of LLMs, highlighting their core capabilities as well as reliability and truthfulness issues that impact their use in complex generation tasks. Section 3.1.3 then outlines the fundamentals of prompt engineering as a discipline for systematically shaping LLM behaviour through input design. It introduces key prompting strategies such as zero-shot, one-shot, and few-shot prompting, as well as broader categories of reusable prompt patterns that support structure, reasoning, and self-evaluation. Additionally, section 3.4 defines the central terms used throughout the thesis, including the concept of Synthetic System Requirement Specifications (SSRS), the structural SSRS template, and the notion of realism as a key quality dimension. These terms provide the conceptual basis for the following chapters.

3.1. Large Language Models

Large Language Models (LLMs) are advanced artificial intelligence systems capable of understanding and generating human-like text. Built primarily on transformer architectures, they are trained on vast datasets from the internet to capture complex linguistic patterns and semantic relationships [Bas+25] [Mai+24]. LLMs excel in tasks such as summarisation, translation, and question answering, making them invaluable in diverse domains, including content creation and research [Hua+24a].

Despite their capabilities, LLMs face challenges like bias, misinformation, and hallucination, where outputs deviate from factual accuracy or fail to remain grounded in input data. Addressing these issues is critical to improving the reliability and trustworthiness of LLMs [Ji+23] [Hua+24a]. This section delves into the strengths and limitations of LLMs.

3.1.1. Capabilities of LLMs

LLMs demonstrate remarkable strengths across a wide range of natural language processing tasks due to their scale, versatility, and ability to generalise from minimal task-specific input. They are capable of performing tasks based solely on a textual prompt that describes the task, without requiring prior training on the specific task at hand . This enables LLMs to achieve strong performance on challenges such as machine translation, question answering, and textual entailment, relying entirely on their pre-trained knowledge and the structure of the input prompt. Moreover, recent work has shown

that their reasoning capabilities can be further enhanced by structuring prompts to include intermediate reasoning steps, which significantly improves accuracy on complex arithmetic and common-sense reasoning tasks . These characteristics make LLMs particularly effective for generative tasks that demand adaptability, contextual awareness, and coherent language generation [Bro+20] [Wei+22] [Qia+22].

3.1.2. Reliability and Truthfulness Issues

While LLMs are powerful tools for generating fluent and contextually relevant text, they can struggle with reliability and factual accuracy. In this thesis, these limitations are grouped under the term *reliability and truthfulness issues*, referring to model behaviours that result in factually incorrect, misleading, or unjustifiable confident outputs. Three of the most critical issues in this area are hallucination, overconfidence, and uncertainty. Understanding and addressing these failure sources is indispensable when trying to generate reliable and truthful outputs using an LLM. In the following, these three issues are described in detail, starting with hallucination.

Hallucination

Hallucination in LLMs is defined as the generation of content that is nonsensical, unfaithful to the input source, or unverifiable. This phenomenon poses significant challenges to the reliability and usability of LLMs, as it undermines their effectiveness in producing accurate, grounded, and contextually appropriate outputs. Addressing hallucination is critical for ensuring the trustworthiness of these models, particularly in applications where factual accuracy is essential [Hua+24a] [Ji+23] [MLG23] [Ton+24].

The impact of hallucination extends across various dimensions. It erodes user trust in LLM-generated outputs, as users may lose confidence in the system's reliability when faced with unfaithful content. Additionally, hallucinations increase the cognitive burden on users, who must validate and potentially correct outputs, diminishing the efficiency gains LLMs are intended to provide. Beyond usability, hallucination has ethical and social implications, as it can propagate misinformation, reinforce biases, or produce harmful content. The need for manual oversight or advanced verification mechanisms to manage hallucination further raises operational costs and limits the scalability of LLMs in broader deployments [Hua+24a] [Ji+23] [Ton+24] [Lia+24].

Hallucination in LLMs can be categorised into two main types. *Intrinsic* hallucination occurs when the generated content contradicts the input source, failing to remain faithful. For example, in summarisation tasks, a model may produce a statement that inaccurately reflects the source material, such as misreporting key dates or events. *Extrinsic* hallucination, on the other hand, involves the inclusion of unverifiable content that is not grounded in the input source. While extrinsic hallucinations may sometimes be factually correct, their lack of grounding introduces uncertainty, especially in contexts requiring strict adherence to source material [Hua+24a] [Ji+23] [Ton+24].

The causes of hallucination can be traced to issues at three key stages: data, training, and inference. *Data-related* causes often stem from mismatches between source and

reference in training datasets, which leads to outputs that deviate from the input. Additionally, heuristic data collection methods can introduce inconsistencies, while duplicate examples in the dataset may result in models over-fitting and generating memorised, irrelevant content. *Training-related* issues include an objective mismatch, where optimisation for fluency leads to degenerated outputs that prioritise linguistic coherence over faithfulness. Furthermore, LLMs often rely excessively on general world knowledge, generating outputs that are not aligned with the specific input. *Inference-related* causes include decoding strategies such as beam search, which tries out several alternative outputs before choosing the best one, and temperature, which adjusts how confident or random the model's word choices are during generation. These methods can sometimes favour outputs that sound plausible or fluent over those that are factually accurate or contextually precise [Hua+24a] [Ji+23] [Ton+24].

Mitigation strategies for hallucination focus on addressing the specific causes identified at each stage. *Data-level* interventions include improved dataset curation to ensure alignment between source and reference, as well as filtering out duplicates and inconsistencies to reduce over-fitting. At the *training level*, task-specific objectives that prioritise input alignment and reinforcement learning with human feedback have shown promising results in reducing hallucination. For inference, tuning decoding parameters, such as beam width and temperature, and employing post-generation editing tools can help detect and correct unfaithful content. These strategies, when combined, provide a comprehensive approach to mitigating hallucination and enhancing the reliability of LLMs [Hua+24a] [Ji+23] [Ton+24] [Lia+24].

While the term black-box is not explicitly defined in the paper by Huang et al. [Hua+24a], the authors describe systems that are "only accessible via API calls" and offer no access to internal structures such as model parameters. Based on this characterisation, this thesis adopts the term *black-box* LLMs, to refer to models with limited accessibility, where internal components cannot be inspected or modified, and the user is only able to observe the outputs. This definition also aligns with common terminology in the overarching field of Artificial Intelligence [Kos24]. Conversely, *white-box* LLMs are defined here as models with accessible internals, such as architecture, parameters, or training configurations. For black-box LLMs such as ChatGPT-4, where data-level or training-level interventions are not feasible, traditional hallucination mitigation strategies are not applicable. In such cases, alternative approaches must be employed to address hallucination. These include post-generation techniques such as output verification using external knowledge sources, leveraging auxiliary models for fact-checking, or employing structured prompts that explicitly constrain the model's output. Additionally, iterative prompting strategies, where the model is asked to self-evaluate and refine its responses, can help improve faithfulness and reduce hallucinations. These methods emphasize adaptability and user-level controls to mitigate hallucination without relying on modifications to the underlying model [Hua+24a] [Lia+24] [MLG23].

Overconfidence

Overconfidence in LLMs is defined as the tendency of these models to present information with high certainty, regardless of whether the content is factually correct or verifiable. This phenomenon poses a significant risk, especially when overconfidence coincides with hallucination, resulting in outputs that are not only incorrect but also expressed confidently. Such outputs can be highly misleading, particularly for users who are non-experts in the domain of the generated content. Unlike systems that are designed to express how certain they are about their predictions, LLMs typically provide their outputs without any indication of how confident they are in their correctness. This is largely because these models are trained to produce text that sounds fluent and coherent, not necessarily to judge whether what they generate is true or reliable [Sun+25] [Tia+23] [Gen+23].

As shown by Sun et al. [Sun+25], LLMs often present incorrect or fabricated information with the same level of confidence as correct information, making it difficult for users to distinguish between the two. The implications of this overconfidence are multifaceted. From a usability perspective, confidently hallucinated outputs reduce a user's ability to detect errors, especially in contexts where domain knowledge is limited or time for verification is constrained. This increases the cognitive burden on users and can lead to the unintentional propagation of misinformation. Furthermore, overconfidence impedes transparency, as the model provides no cues to distinguish between reliable and potentially erroneous content. In safety-critical or high-stakes applications, this behaviour can significantly undermine trust in the system. Without mechanisms to reveal uncertainty or self-reflect on output quality, users may develop misplaced confidence in model-generated information [Sun+25] [Gen+23].

Efforts to mitigate overconfidence in LLMs include confidence calibration, uncertainty-aware prompting, and response justification techniques. These approaches aim to align a model's expressed certainty with the actual reliability of its outputs. In scenarios where access to the model's internals is restricted, strategies such as self-assessment prompts or structured output formats may help expose uncertainty and reduce the misleading impact of overconfident responses. In summary, overconfidence is a critical failure mode that amplifies the risks associated with hallucination and hinders error detection. Addressing this issue is essential for improving the reliability, transparency, and usability of LLM-generated outputs [Sun+25] [Xio+23] [Sch+24] [Tia+23] [Hua+24b] [Gen+23].

Uncertainty

Uncertainty in LLMs refers to the variability or inconsistency in outputs given the same input, caused by ambiguous inputs, model limitations, or the open-ended nature of the task. Uncertainty is an inherent characteristic of LLMs, impacting their reliability and interpretability. Understanding the sources and types of uncertainty is essential to effectively quantify and manage it in LLM-generated outputs. Uncertainty in LLMs arises from multiple sources. At the *input level*, ambiguity in user inputs, such as complex sentence structures, domain shifts, or out-of-vocabulary words, can introduce uncertainty into the model's predictions. At the *system level*, the architecture of the

model, parameter initialisation, and training processes contribute variability that affects uncertainty in outputs. Lastly, at the *output level*, uncertainty is amplified in generative tasks that require a degree of creativity or the generation of entirely new content, such as story generation or open-ended question answering. In such tasks, generating longer sequences or intricate outputs tends to increase uncertainty, making it harder to ensure the consistency and reliability of the results [Hu+23] [Hou+23] [Sho+24].

In addition to these sources, uncertainty can be further categorised into two types: *aleatoric* uncertainty and *epistemic* uncertainty. Aleatoric uncertainty arises from noise or ambiguity in the input data itself, such as unclear or contradictory information provided to the model. In contrast, epistemic uncertainty reflects gaps in the model's knowledge or its inability to generalise effectively due to insufficient or biased training data. While aleatoric uncertainty is often irreducible, epistemic uncertainty can potentially be reduced by improving the training process or expanding the training dataset. By understanding and addressing both the sources and granularity of uncertainty, LLMs can be better equipped to handle ambiguous inputs and produce reliable outputs, ensuring they meet the needs of their intended applications [Hou+23] [Hu+23] [Sho+24].

3.1.3. Prompt Engineering

Understanding the challenges posed by hallucination, overconfidence, and uncertainty is essential for the responsible use of LLMs. Yet recognising these limitations is only the first step. Actively mitigating them requires methods for influencing and controlling model behaviour. One of the most prominent approaches is encompassed in the discipline of prompt engineering. *Prompt engineering* refers to the systematic design, structuring, and refinement of prompts to improve the accuracy, reliability, and task alignment of LLM outputs. Rather than modifying the model itself, prompt engineering works by shaping the input to elicit more desirable responses. It has gained traction as a practical approach to enhance LLM performance, especially in cases where retraining or fine-tuning is infeasible. In this thesis, prompt engineering serves as a central mechanism for addressing the reliability issues outlined above and for enabling high-quality, domain-specific generation without relying on external ground truth or expert supervision. Prompt engineering is particularly important for applications that utilise black box LLMs, because in these cases users can only control the behaviour of the LLM through their inputs. Thus writing high-quality prompts is essential and should be supported by prompt engineering techniques to improve output quality [Sch+24] [Whi+23] [Wei+22].

Prompt engineering encompasses a variety of methods for structuring input. One foundational subset of these methods is *Prompting Strategies*, which define the general approach of how to accurately communicate a task to an LLM to achieve a certain desired output. Another important concept is a group of techniques called *Prompt Patterns*, which define reusable design principles for guiding LLM behaviour. The following sections present both concepts in more detail.

3.2. Prompting Strategies

Prompting strategies are overarching approaches used to structure how a task is communicated to an LLM. These strategies define how much context or guidance is provided in the input, and play a central role in shaping the model's ability to produce accurate and relevant outputs for a given task. Different prompting strategies vary in how explicitly they define the task, whether through input-output examples, direct instructions, or both. There are three foundational strategies: zero-shot, one-shot, and few-shot prompting. In the following each approach is introduced [Sch+24] [Bro+20].

Zero-shot prompting is defined as presenting the model with only an instruction or task description, without any examples of correct outputs. This approach relies entirely on the model's internal knowledge and generalization capabilities. It is particularly effective when tasks are either well-known (e.g., language translation or fact-based questions) or open-ended and exploratory in nature (e.g., creative generation), where examples may constrain the model unnecessarily or introduce bias. Zero-shot prompting is also the only option in cases where a ground-truth is simply not available or hard to establish, and thus no evaluated examples can be provided to the LLM [Sch+24] [Bro+20].

Few-shot prompting is defined as providing the model with a small number of carefully selected input-output examples to help it infer both the structure and semantics of the task. This strategy is well-suited for more complex or unfamiliar tasks where the desired format or reasoning process is difficult to convey through instructions alone. However, it assumes the availability of representative examples, which may not always be feasible, especially in novel or data-scarce domains [Sch+24] [Bro+20].

One-shot prompting sits between the two extremes. It includes a single example of the desired input-output pair to demonstrate the task format or intent. This strategy is helpful when a minimal example can clarify ambiguous or under-defined instructions while avoiding the complexity associated with few-shot prompts [Sch+24] [Bro+20].

While prompting strategies define foundational approaches of how to communicate a task to an LLM, they do not specify how the actual task description is implemented. The following section thus introduces the concept of prompt patterns, which can be used to construct task descriptions based on scientifically evaluated design principles.

3.3. Prompt Patterns

While the prompting strategies presented above are foundational for guiding LLM behaviour, their effectiveness can often be enhanced through the application of structured, reusable design principles. One such approach within the broader field of prompt engineering is the use of prompt patterns. A *Prompt pattern* is a generalisable template or heuristic for constructing prompts that is effective across tasks and domains, very similar to software design patterns, which offer proven solutions to common design problems in programming. In both domains, patterns encapsulate best practices into modular structures that promote consistency, adaptability, and efficiency. In the context of LLMs, prompt patterns are defined as task-agnostic design structures that specify a particular

way of framing input or instructions to guide the model's response predictably and effectively. The patterns are not tied to specific tasks or domains, making them broadly applicable in prompt formulation. [Whi+23] [Sch+24].

3.3.1. General Purpose Patterns

General purpose prompt patterns are a category of reusable prompt structures designed to improve output quality across a wide variety of tasks and domains. Unlike more specialised patterns that target specific problem types, general purpose prompt patterns aim to enhance clarity, structure, and alignment in model responses regardless of the task at hand. These patterns typically focus on how instructions are framed or how the model is positioned in the prompt. These techniques may involve assigning the model a specific role to help it generate more contextually relevant responses, or they may define the expected output format explicitly to ensure structural consistency. One common approach is the persona pattern, which assigns the model a specific persona or role, such as a software engineer or security expert, to guide its tone, expertise, and perspective in generating more contextually appropriate responses. The primary advantage of general purpose patterns lies in their versatility. Because they do not rely on task-specific logic or assumptions, they can be applied in a broad range of use cases. As such, they form a foundational toolset for prompt engineers looking to improve model performance through better prompt design [Whi+23] [Sch+24].

The specific general purpose patterns applied in this thesis are introduced in chapter 5.

3.3.2. Structured Reasoning Patterns

While general purpose prompt patterns offer flexible tools for shaping model behaviour and structuring outputs, complex tasks often require more targeted strategies to support reasoning and decision-making. *Structured reasoning* patterns are a category of prompt design techniques that guide a language model through a complex task by explicitly breaking it down into intermediate steps or sub-problems. Rather than asking the model to solve a complex prompt in a single pass, these patterns encourage incremental reasoning, which has been shown to improve coherence, factual accuracy, and task success rates in multi-step or logic-intensive scenarios. As an example, the Chain-of-Thought pattern in its simplest version prompts an LLM to "think step-by-step" to encourage step-wise reasoning for a given task. These patterns are particularly valuable in domains where responses require justification, sequencing, or dependencies between different aspects of the output. They help mitigate common LLM failure modes such as hallucination, inconsistency, or shallow reasoning by encouraging the model to articulate intermediate reasoning steps explicitly [Sch+24] [Wei+22] [Zha+22].

The specific structured reasoning patterns investigated in this thesis are introduced in chapter 5.

3.3.3. Self-Criticism Patterns

After highlighting key prompt engineering techniques to support LLMs in output generation and task solving, this section turns to techniques applied after the initial output generation. *Self-Criticism* is a category of prompt patterns encompassing all methods where an LLM is instructed to assess and potentially refine its own outputs. Self-Criticism techniques can be further divided into two main types: *Self-Assessment*, where the model provides a qualitative or quantitative judgment of its own output, and *Self-Refinement*, where the model not only critiques but also refines the original response based on its own evaluation [Sch+24] [Li+24] [Wei+24] [Mad+23].

All prompt patterns selected and applied in this thesis are discussed in detail in chapter 5.

While the previous sections provided conceptual tools for structuring and evaluating model behaviour, the following section defines the key terminology that will be used throughout the remainder of this thesis.

3.4. Terminology

This section defines the essential terminology that forms the basis for the thesis's main artefacts and evaluation criteria. It begins by defining the central artefact generated in this work, along with the template that specifies its structure and content. This is followed by a definition of the term *Realism*, which serves as the conceptual basis for one of the core quality criteria used to evaluate the generated artefacts.

3.4.1. Synthetic System Requirement Specification

Definition

A *Synthetic System Requirement Specification (SSRS)* is a structured, domain-specific artefact that represents a natural-language description of a hypothetical software system. Its content is adapted from the structure and terminology defined in ISO/IEC/IEEE 29148:2018 [ISO18] for formal System Requirements Specifications (SyRS).

The adapted SyRS used in this thesis comprises four main sections with the following content:

1. **System Overview:**
   - *System Purpose* – High-level description of the system's main objectives and goals.
   - *Domain/Context* – The domain or industry in which the system operates.
   - *Stakeholders* – Key individuals or groups that interact with or are affected by the system (e.g., users, administrators, regulators).

- *User Base Characteristics* – Size, diversity, geographic distribution, and roles (e.g., employees, customers, administrators).
- *Operational Environment* – Where and how the system is hosted or accessed (e.g., cloud-based, on-premise, mobile).
- *Usage Scenarios* – Common day-to-day user interactions or workflows.

2. **Functional Requirements:**
   - *Core Features* – Specific functionalities the system must provide (e.g., data input, process automation).
   - *Authentication Conditions & Frequency* – How often users need to authenticate and under what circumstances (e.g., session expiration, sensitive actions).
   - *Sensitivity of Actions & Permission Levels* – Operations requiring authentication, their sensitivity levels, and associated user roles or permissions.
3. **Non-Functional Requirements:**
   - *Performance* – Expected metrics such as response times and system throughput.
   - *Scalability* – Ability to handle increases in users or data without degradation in performance.
   - *Reliability* – Expectations for fault tolerance, error handling, and recovery mechanisms.
   - *Security* – Measures to ensure data protection and access control.
   - *Usability* – Requirements for user accessibility and ease of interaction.
   - *Audit & Monitoring* – Requirements for tracking, logging, and reviewing authentication events or user actions.
4. **Constraints:**
   - *Technical Constraints* – Hardware, software, or infrastructure limitations.
   - *Compliance Requirements* – Legal, regulatory, or domain-specific obligations (e.g., GDPR, HIPAA).
   - *Resource Constraints* – Budget, staffing, or time limitations affecting development or operation.
   - *Integration Needs* – Whether the system must integrate with existing authentication systems or infrastructure.

The structure and contents of each section are selected to fit the specific requirements of evaluating the SecuRe recommender system, therefore, detailed information on security requirements was added. In this context, System Requirements Specifications (SyRSs) serve as a source from which realization contexts, as described in section 2.2, can be extracted.

3.4.2. SSRS Template

Definition

The *SSRS Template* is an instantiation of the template prompt pattern, which defines the content of SSRSs in the context of this thesis. It specifies all elements together with an explanation or example. The SSRS template is instantiated with content tailored to specific industry domains.

An *Industry Domain (IndDom)* denotes a distinct sector such as finance, healthcare, or logistics and in the following will be abbreviated with *IndDom*.

While the SSRS template utilised in this thesis is tailored specifically to the evaluation of the SecuRe constraint-based recommender system, the underlying concept is not tied to this application context. The SSRS template was adapted from ISO 29148 to address the specific evaluation needs of the recommender system, incorporating detailed information on authentication requirements. This adaptation was desired as, at the time of writing this thesis, SecuRe focuses exclusively on authentication security patterns. Thus, the approach can similarly be transferred to other application contexts by tailoring the SSRS template to their respective evaluation needs.

3.4.3. Realism

In addition to defining the structural template for SSRS instances, it is also necessary to clarify how the quality of their content is conceptually framed. One key aspect in this context is the notion of realism, which serves as a guiding principle for evaluating the plausibility of generated requirements. In the context of this work, the term *realism* is understood as *"representing things in a way that is accurate and true to life"* [Oxf23]. This notion of realism is based on the definition for the word *realistic* in the Oxford English Dictionary. Realism is a crucial aspect in guiding the evaluation of SSRSs, focusing on whether the generated requirements plausibly reflect real-world system requirements within their respective industry domains.

After introducing the foundational terminology, the following chapter presents the methodology applied to design a structured process to generate realistic SSRS based on the pre-defined SSRS template.

4. Methodology

This chapter outlines the methodological basis used to inform the design of a structured, LLM-based process for generating IndDom-specific Synthetic System Requirements Specifications (SSRSs). Rather than starting from an arbitrary implementation, the approach was grounded in a targeted review of existing literature. Two focused literature reviews were conducted to support the design. The first one reviewed prompt engineering strategies aimed at improving the quality and control of LLM-generated outputs. The second one examined evaluation techniques suitable for assessing generated content in scenarios where ground-truth data is scarce or unavailable.

Building on the insights from these reviews, a subsequent prototyping phase was undertaken to iteratively design, validate, and refine a structured process for generating realistic SSRSs. To support consistent terminology throughout this thesis, the term SSRS-Gen process is introduced. *Synthetic System Requirement Specification Generator (SSRS-Gen)* refers to the LLM-based framework developed in this work to produce realistic and structured SSRSs. The process was designed to be domain-independent, scalable, and capable of producing outputs with high structural and semantic quality.

The following section defines the key requirements that shaped the design space of the SSRS-Gen process. These requirements served as the foundation for selecting, adapting, and combining techniques during the development phase.

4.1. Process Requirements

The SSRS-Gen process was developed to address the need for generating realistic SSRS in zero-resource settings. To guide its design, a set of six core requirements was defined. Each requirement is clearly defined and explained below.

R1 – Realism: *The process must generate SSRSs that plausibly resemble real-world system requirement specifications.*

Realism is essential for enabling SSRSs to function as meaningful substitutions for real-world SyRS in domains where access to this data is limited or unavailable. Outputs must be realistic to be usable for any evaluation or testing purposes.

R2 – Comparability: *The process must produce SSRSs with a consistent structure and content coverage, enabling meaningful comparison across multiple outputs.*

To support objective evaluation, reuse, and downstream analysis, all generated SSRSs must conform to a predefined content structure. This structure not only enforces struc-

tural consistency but also defines the expected types of content.

R3 – Diversity: *The process must generate multiple SSRSs within the same industry domain that are not redundant, but instead represent distinct and meaningful variations.*

Diverse outputs within the same industry domain increase the utility of the generated SSRSs, particularly for use in testing and evaluation task. Without diversity, the value of generating multiple SSRS diminishes.

R4 – Domain Independence: *The process must support generation across multiple industrial domains without relying on domain-specific fine-tuning or prior examples.*

This requirement was driven by the intended use of SSRSs in evaluating the SecuRe recommender system, which requires a broad range of realization contexts, as introduced in section 2.2. As such, SSRS-Gen must be capable of producing SSRSs for a set of distinct industry domains, without requiring task-specific adaptation or customization.

R5 – Zero-Resource Operation: *The process must function in settings where no labelled datasets or existing SyRS documents are available.*

This requirement acknowledges the scarcity of high-quality, publicly available SyRS documents. Moreover, even when such documents are accessible, their quality cannot be guaranteed without expert validation, which introduces additional complexity and bias risk. To ensure applicability in data-constrained environments, the process must operate without reliance on any external data sources.

R6 – Expert Independence: *The process must operate without the involvement of domain experts at any stage, including generation, evaluation, and refinement.*

Relying on expert feedback across multiple iterations would make the process infeasible due to the high workload involved. Expert independence ensures that SSRS-Gen remains scalable, practical, and applicable in environments where experts are rare or not available.

These six requirements form the foundation for the design and scope of the SSRS-Gen process. Together, they define the constraints under which the process must operate and the qualities that its outputs must satisfy. In particular, the emphasis on zero-resource conditions and expert independence significantly narrows the space of viable methodological approaches. To inform the design of SSRS-Gen under these constraints, a targeted literature review was conducted to identify existing methods in prompt engineering that could operate within these boundaries.

4.2. Targeted Literature Review

To inform the design of SSRS-Gen under the constraints defined in the previous section, a targeted literature review was conducted to identify generalisable methods, prompting strategies, and evaluation techniques focused on prompt engineering and the assessment of LLM-generated content. This review provided a pragmatic and empirically grounded foundation for the design of a novel generation process aligned with the thesis objectives. By drawing on techniques with demonstrated value, the resulting process aims to reliably produce high-quality SSRSs, even in domains where access to real-world SyRS or continuous expert feedback is limited.

The first literature review investigated prompt engineering strategies and prompt patterns. Its objective was to identify techniques capable of improving the quality, structure, and domain alignment of LLM-generated outputs in complex text generation tasks. Special attention was given to strategies aimed at mitigating common limitations of LLMs, such as hallucinations, overconfidence, and uncertainty, as discussed in section 3.1.2. The result was a curated list of prompt patterns and engineering techniques that served as the conceptual foundation for subsequent process design and refinement.

The second literature review focused on evaluation methods for assessing the quality of LLM-generated text in contexts where no ground truth is available. It explored existing automatic and model-based metrics, as well as more recent LLM Self-Assessment approaches. The review emphasized the need for task-specific metrics that could evaluate SSRS outputs with respect to the three requirements of *R1-Realism*, *R2-Comparability* and *R3-Diversity*, as these define explicit quality criteria for generated SSRS. Together, these two reviews established the methodological basis for the design of the SSRS-Gen process and directly informed the decisions made in subsequent chapters regarding both the design principles and evaluation strategies used.

It is important to note that the literature review conducted was not intended to be exhaustive in a systematic review sense. Instead, a targeted manual search strategy was employed, focusing on literature describing empirically validated prompting strategies and evaluation techniques. The selection emphasized practical applicability and conceptual relevance over comprehensive coverage, aligning with the exploratory and design-oriented goals of this thesis.

The following section outlines how the findings from the literature reviews informed the prototyping of the SSRS-Gen process.

4.3. Process Prototyping

Based on the insights derived from the targeted literature reviews, a prototyping phase was initiated to validate the feasibility of a structured generation process for SSRSs subject to the given constraints. In this context, prototyping refers to the iterative design and partial testing of the process components from scratch, aimed at evaluating whether the approach fulfilled its intended requirements. Assessing output quality was not part of the prototyping phase itself, as quality control was integrated into the process through

its inherently iterative structure. The central objective of this phase was to design a process capable of generating high-quality SSRSs without access to real-world SyRS or the involvement of human domain experts, as both resources are typically scarce as highlighted before. The prototyping approach was driven by the goal of meeting the core quality criteria for generated SSRSs of comparability, realism, and diversity, while adhering to the process constraints. Accordingly, methods were selected from the literature that could be operationalised in a fully automated LLM setting. This led to the inclusion of prompting strategies to enhance generation quality and self-assessment techniques to enable the LLM to evaluate outputs in the absence of human feedback or ground-truth references.

Initial process design began with a minimal and logically straightforward structure. A single SSRS would be generated, followed by a sequence of evaluations targeting each of the three quality dimensions. This sequence for a single SSRS formed the foundational unit of the process. However, to align with the broader objective of producing diverse SSRSs across multiple industrial domains, the scope was extended. The expanded process was conceptualized around two parameters, m as the number of distinct IndDoms, and n the number of SSRSs to be generated per IndDom. These parameters were directly derived from the core requirements established in in the previous section 4.1. Specifically, the inclusion of multiple IndDoms (m) addresses *R4–Domain* Independence, which was motivated by the need to generate SSRSs across varied domains for the evaluation of the SecuRe recommender system. Meanwhile, the parameter n with $n > 1$ for each IndDom enables the evaluation of *R3–Diversity*, as multiple outputs are required within the same domain to assess whether the generated SSRSs exhibit meaningful variation rather than redundancy. The structure of this multi-domain generation framework was first formalized through conceptual diagrams and then tested in a single-domain setting to verify basic functionality. This initial proof of concept focused on evaluating whether the envisioned sequential workflow for generating multiple SSRSs within a single domain was practically feasible.

The prototyping phase outlined above established a conceptual and operational foundation for the SSRS-Gen process, grounded in the practical constraints and quality goals identified at the outset of this thesis. To move from this preliminary framework toward a fully specified process design, it was necessary to translate the high-level process logic into concrete prompting and evaluation strategies. The following chapter presents the results of the targeted literature reviews in detail, outlining the specific techniques identified and the rationale behind their selection for the process. These findings directly informed the design choices underlying the final SSRS-Gen process and serve as the empirical basis for its prompting and assessment components.

5. Selected Prompting Techniques and Evaluation Metrics

This chapter presents the results of the two targeted literature reviews that informed the design of the SSRS-Gen process. The first review focused on identifying prompt engineering techniques, specifically prompt patterns, that have demonstrated effectiveness in guiding LLM behaviour, improving output quality, and mitigating known reliability and truthfulness issues, as introduced in section 3.1.2. The second review addressed evaluation methods for LLM-generated content, with a focus on identifying metrics that can assess output quality in contexts where ground-truth references are unavailable. Together, these two reviews provide the empirical and conceptual foundation for selecting the techniques and evaluation criteria applied throughout this thesis. The findings are grouped into three main sections: prompting strategies, followed by prompt patterns, and lastly task-specific evaluation metrics tailored to the requirements for SSRS generation.

5.1. Prompting Strategies

The design of the SSRS-Gen process began with the fundamental challenge of how to accurately communicate the desired output structure and content to an LLM. This required not only a precise definition of the intended SSRS format, but also a well-formulated prompt, capable of consistently eliciting outputs that conformed to this specification. As introduced in section 3.2, prompting strategies define the fundamental approach used to instruct an LLM to perform a given task. Based on the three presented core strategies it had to be evaluated which prompting strategy is the most suitable for this work. In the context of this thesis, few-shot prompting would necessitate one or more high-quality SSRS examples created and evaluated by IndDom experts to serve as output examples. However, access to both real-world SyRS and qualified IndDom experts is limited, which presents a critical barrier for few-shot prompting, as the approach relies on representative, high-quality examples to guide generation. Moreover, incorporating unevaluated or biased examples into the generation pipeline would compromise the validity of the process, as the model could internalise and amplify these flaws, leading to systematic errors in output generation and masking the effects of subsequent prompt refinements intended to improve quality. These constraints render the few-shot approach infeasible in the context of this thesis. Instead, a zero-shot prompting strategy was adopted. To support this strategy and ensure that the model could still reliably produce structured and contextually appropriate outputs, the following sections present the prompt pat-

terns identified during the literature review as most suitable for guiding generation in the absence of examples.

5.2. Prompt Patterns

As outlined in section 3.3, prompt patterns offer structured, reusable approaches to frame instructions that can enhance the quality and reliability of LLM outputs. Given that few-shot prompting was not feasible for this thesis due to the lack of available example data, prompt patterns became a central mechanism for communicating the desired output structure and intent in a zero-shot setting. They provide a principled way to guide model behaviour without relying on task-specific examples, making them particularly well-suited for the SSRS generation context [Whi+23] [Sch+24].

The following sections present the specific prompt patterns identified through the literature review as most relevant for supporting the zero-shot generation of SSRSs.

5.2.1. General Purpose Patterns

This subsection presents general-purpose prompt patterns identified in the literature that are applicable across a wide range of tasks. These patterns were selected for their potential to address core challenges of the SSRS-Gen process, including format consistency, contextual realism, and zero-shot applicability.

Template Pattern

The *Template pattern* is a prompt engineering technique where the user specifies the desired output format and content type through a template, which is then filled with actual content by the LLM. The template is often defined using structured text containing placeholders for the LLM to fill with information in the final output. This pattern is useful in applications where the LLM output is used in any subsequent tasks that require a specific input format, or in cases where the same template-based prompt is executed multiple times. In the latter case, the template pattern supports comparability between outputs of the same prompt as it ensures consistency in output format [Sch+24] [Whi+23].

In the context of the SSRS-Gen process, the template pattern was selected as a core mechanism for defining the desired structure of generated outputs. As introduced in section 3.4, the SSRS template specifies the expected content and format. The use of this pattern was motivated by two main factors. First, the process has to rely on zero-shot prompting, which precludes the use of SSRS examples to convey output format. The explicit template thus provides the necessary structural guidance to the model. Second, structural consistency across outputs was considered essential to support systematic evaluation and iterative refinement. The template pattern was therefore selected based on the assumption that explicitly specifying the desired structure, defined by the SSRS template, would help elicit outputs that adhered to this format. This, in turn, was

expected to support the process requirement of *R2-Comparability* as introduced in section 4.1, and therefore enabling a meaningful analysis across domains and prompt versions.

Persona Pattern

The *persona pattern* is a textual instruction that prompts an LLM to adopt a specific role or viewpoint in generating its output. This persona can represent a profession, a domain expert or even a fictional character. The pattern intends to shape the model's perspective in producing responses that reflect, most importantly, the knowledge, style and decision-making typical of the specified role. For instance, instructing the LLM to "act as a security expert" leads it to emphasise security-focused observations in code reviews. This pattern is particularly useful when users know the type of perspective or expertise needed for a task but may lack the technical detail to specify the desired output directly themselves. This also applies to the context of this thesis, as the goal is to generate IndDom-specific SSRS in the absence of human experts that could design output examples or a ground truth. By defining a relevant persona, the LLM is prompted to generate contextually aligned and detailed outputs. The effectiveness of this pattern depends on how accurately the persona is defined with regard to the task the LLM should perform and the underlying domain context [Whi+23] [Sch+24].

Based on the insights of the literature review, the persona pattern was selected for the planned use in the SSRS-Gen process as a means to enhance the contextual depth and domain relevance of the generated SSRS. In the context of the chosen zero-shot prompting strategy, the persona pattern was assumed to help compensate for the lack of detailed SSRS examples by prompting the model to adopt the perspective of a relevant domain expert. This, in turn, was expected to increase the level of detail and realism in the generated requirements, particularly in relation to the given IndDom.

5.2.2. Structured Reasoning Pattern

Structured reasoning techniques are particularly relevant in this thesis because the generation of SSRSs involves filling multiple interdependent sections of a predefined template with realistic and logically coherent content. To support the model in handling this complexity, the literature review identified three structured reasoning patterns applied in prompt engineering: *Chain-of-Thought*, *Tree-of-Thought*, and *Least-to-Most* prompting. Each of these strategies encourages the model to approach complex tasks in a stepwise fashion but differs in how intermediate steps are generated and utilised.

Chain-of-Thought prompting is a structured reasoning approach that instructs an LLM to explicitly generate a sequence of intermediate reasoning steps before producing a final answer. It can be applied as a zero-shot approach by using general cues such as "think step-by-step" or by including the specific sub-steps of a given task and prompting the LLM to solve these sub-tasks sequentially before arriving at a final solution. The latter approach can also be adapted to a few-shot version, by demonstrating the specific sub-steps using concrete examples, for instance by showing all solving steps of a mathematical problem. Chain-of-Thought prompting is particularly effective for tasks that require

multi-step reasoning, such as mathematical or logical problem solving, but also commonsense reasoning or planning and decision-making tasks. By additionally prompting the LLM to output all intermediate steps, this technique can also be used for getting insights into how an LLM arrives at solutions for a given task, which can help to more easily identify flaws in its reasoning process [Sch+24] [Wei+22] [Zha+22].

Among the three identified structured reasoning patterns, *Chain-of-Thought prompting* was selected for planned use in the SSRS-Gen process. This pattern was selected due to the inherent complexity of generating SSRSs: each section of the output must be populated with realistic, domain-specific content while maintaining logical coherence across sections, as defined by the SSRS template. Literature on prompt engineering indicates that chain-of-thought prompting can help LLMs break down intricate tasks into smaller, more manageable sub-problems, thereby improving output coherence and contextual plausibility [Sch+24] [Wei+22] [Zha+22].

In contrast, *Tree-of-Thought prompting* was not selected due to its significantly higher implementation complexity. Tree-of-Thought extends Chain-of-Thought reasoning by exploring multiple reasoning paths in parallel and evaluating them to choose the most promising solution branch. Applying it in the SSRS context would require the model to generate and evaluate multiple alternative instantiations for each section of the SSRS template, followed by a selection step to determine the most suitable path. This process would not only be time-consuming but would also require the model to evaluate the quality of these instantiations itself and select the best option. It was assumed that delegating the evaluation and selection process to the LLM could introduce uncontrolled biases into the generated SSRS, as only the final selected output would be available for review, making it difficult to trace or assess the influence of discarded alternatives [Sch+24] [Yao+23].

Similarly, *Least-to-Most prompting* was excluded because of its complexity. Least-to-Most prompting is a technique where the model is instructed to first solves simpler sub-problems before using those intermediate solutions to tackle a more complex, overarching problem. This approach of first solving each sub-task independently would likely result in logically incoherent outputs. Since each SSRS section must align logically with the others, generating them in isolation and merging them afterwards poses a substantial risk of internal contradictions. Chain-of-Thought prompting offered a more pragmatic balance between reasoning support and structural coherence, making it the most appropriate choice within the scope of this thesis [Sch+24] [Zho+22].

5.2.3. Self-Criticism Patterns

After highlighting key prompt engineering techniques to support LLMs in output generation and task solving, this section turns to Self-Criticism patterns, as introduced in section 3.3. In the following the Self-Assessment pattern is presented in detail along with the rationale for its selection for the SSRS-Gen process.

Self-Assessment is a category of prompting techniques encompassing all methods where an LLM is instructed to evaluate or critique its own previously generated output. This internal evaluation may involve scoring the output's quality, ranking it against

alternatives, or commenting on aspects such as correctness, coherence, or factual accuracy. In this thesis, self-assessment is employed as a mechanism for evaluating the quality of generated SSRSs without requiring external ground truth or expert review. Prior research has shown that LLMs can conduct self-assessments with a degree of sensitivity and semantic nuance that surpasses traditional automatic metrics. For instance, Li et al. [Li+24] demonstrate how prompting a model to reflect on its own outputs can yield useful quality judgments through scoring or ranking strategies. This is especially valuable in open-ended tasks where reference-based evaluation is infeasible. However, self-assessment techniques also exhibit inherent limitations that must be carefully considered when applying them in practice. As Wei et al. [Wei+24] point out, LLMs can exhibit biases such as favouring longer responses, showing sensitivity to output order, and producing inconsistent evaluations across repeated assessments. Despite these challenges, self-assessment remains a scalable and practical approach for quality estimation in LLM-driven workflows, provided that the prompting is carefully designed to minimize known biases and maximize consistency [Li+24] [Wei+24].

For generating and evaluating SSRS in this thesis the Self-Assessment prompt pattern was chosen. This approach was selected to enable automation and scalability without requiring continuous involvement from human industry domain experts. Relying on expert reviewers would have substantially limited the feasibility of an iterative refinement process and contradicted the overarching goal of developing a methodology suitable for settings in which real-world SyRS documents and qualified expert feedback are scarce or unavailable. While using a separate model for evaluation might improve the independence of the assessment process, this option was not pursued due to concerns about inconsistent domain understanding, greater operational complexity, and increased computational cost. Self-assessment was therefore adopted as a practical and internally consistent mechanism to maintain reproducibility, reduce human involvement, and support the refinement loop throughout SSRS-Gen. The inclusion of a self-refinement step was not pursued, as the accuracy of self-assessments for complex and domain-specific artefacts such as SSRSs could not be reliably assessed in advance. Incorporating potentially flawed self-assessments into the generation process risked reinforcing inaccuracies and further degrading the overall quality of the output.

Together, the identified prompting strategies and selected prompt patterns form the conceptual foundation for the design of the SSRS-Gen process. These techniques were selected based on their demonstrated potential to improve the quality, specificity, and coherence of LLM-generated outputs. With this prompting foundation established through the first literature review, the next step was to define appropriate evaluation mechanisms capable of assessing the generated SSRSs against the requirements of *R1-Realism*, *R2-Comparability* and *R3-Diversity* as introduced in section 4.1.

The following section presents the findings of the second literature review, which focused on identifying suitable evaluation metrics in the context of missing ground-truth references.

5.3. Evaluation Metrics

Evaluating the outputs of LLMs is essential to assess their quality, reliability, and suitability for specific tasks. In the context of text generation, evaluation metrics serve as instruments to quantify different dimensions of output quality. Evaluation metrics for text generation can be broadly categorised into reference-based and reference-free approaches. Reference-based metrics evaluate generated content by comparing it to one or more predefined ground-truth references. These metrics typically focus on surface-level lexical overlap or shallow syntactic similarity. Common examples include BLEU [Pap+02], which measures n-gram precision; ROUGE [Lin04], which emphasises recall over overlapping sequences; and METEOR [BL05], which incorporates stemming and synonym matching. These metrics have been widely used in machine translation, summarisation, and other natural language generation tasks where high-quality reference texts are available. However, such metrics often fail to capture deeper semantic similarity or to function effectively in open-ended tasks without a reliable reference. In response, more recent methods have adopted embedding-based metrics, such as BERTScore [Zha+19a] and MoverScore [Zha+19b], which compare contextualised representations of the texts rather than their surface forms. Additionally, learned evaluation metrics such as BLEURT [SDP20] and COMET [Rei+20] are trained on human judgment data and better align with human preferences in evaluating generation quality. Finally, a growing class of reference-free methods leverage LLMs themselves to assess output quality. These Self-Criticism methods, as introduced in section 5.2.3, allow LLMs to reason about their own outputs directly without relying on a ground-truth.

To address the specific demands of SSRS generation, this thesis defines three custom evaluation metrics tailored to the quality dimensions most relevant for this task. Standard metrics such as BLEU, ROUGE, or BLEURT were not adopted, as they either assume the availability of a ground truth or prioritise surface-level similarity, which is insufficient for capturing the structural and semantic fidelity required in SSRS evaluation. Instead, the following task-specific metrics are introduced: *Completeness*, *Degree of Realism*, and *Semantic Similarity*. These metrics are formally defined in the subsequent sections.

5.3.1. Completeness

Completeness is a structural evaluation metric used to assess whether a generated output fully conforms to an expected format or schema. In the context of text generation, this typically involves verifying the presence of all required components or sections as defined by an external standard, template, or prompt instruction. In this thesis, Completeness addresses the process requirement of *R2–Comparability* introduced in section 4.1, ensuring that all generated SSRSs follow a uniform structure suitable for systematic evaluation. Specifically, it assesses whether each SSRS contains every element defined in the SSRS template, as introduced in section 3.4. The template is implemented using the template prompt pattern described in section 3.3, which explicitly outlines the expected content and section structure. Completeness is assessed using a binary value: an SSRS

is classified as *true* (complete) if all required elements defined by the SSRS template are present, and as *false* (incomplete) if even a single element is missing. This strict interpretation reflects the structural nature of the Completeness criterion, and unlike more subjective quality attributes, it does not admit partial fulfilment. Since the underlying purpose of this metric is to guarantee comparability across SSRSs, any deviation from the defined structure would undermine their usefulness in subsequent evaluation or testing contexts. As such, the binary formulation is not merely a design choice but a direct consequence of the definition and intended function of the metric.

5.3.2. Degree of Realism

Degree of Realism (DoR) is a qualitative evaluation metric that captures the extent to which a generated output plausibly resembles something that could exist in a real-world context. Unlike structural metrics, which operate in binary terms, DoR assesses more subjective qualities such as plausibility, coherence, and domain alignment. As such, it represents a continuous and interpretable quality dimension that admits partial fulfilment. In the context of this thesis, DoR measures how accurately a generated SSRS reflects real-world SyRS within its intended industry domain and thus directly maps to the defined process requirement of *R1–Realism*, as defined in section 4.1. This includes the credibility of the described system, the feasibility of its stated requirements, and the alignment with commonly accepted practices and constraints within the domain. The metric directly reflects the concept of *Realism* as defined in section 3.4.3, where realism is understood as the extent to which an SSRS aligns with domain-specific expectations and resembles documents that would plausibly be authored by human experts. DoR therefore serves as a critical indicator of whether synthetic SSRSs can act as effective substitutes for real specifications in evaluation and testing contexts where ground-truth data is unavailable.

While DoR captures the plausibility of individual SSRSs, it does not account for redundancy or variation across multiple outputs. However, in order for the generation process to be practically useful, it must not only produce realistic SSRSs, but also ensure that these are meaningfully distinct within the same IndDom. This dimension of quality is addressed by the third metric: *Semantic Similarity*.

5.3.3. Semantic Similarity

Semantic Similarity is a quantitative evaluation metric that measures the degree of semantic equivalence between two textual artefacts, independent of exact wording or surface-level structure. Unlike structural metrics such as *Completeness*, or qualitative metrics such as *Degree of Realism*, this metric is comparative in nature, relying on pairwise analysis of outputs to identify redundant or overly similar content. In the context of this thesis, Semantic Similarity is used to evaluate the extent to which multiple SSRSs generated for the same IndDom are semantically distinct and thus directly links to the process requirement of *R3–Diversity*, as presented in section 4.1. The underlying assumption is that SSRSs intended to support evaluation and testing tasks must not

only be complete and realistic, but also diverse. Generating several SSRSs that express nearly identical requirements would diminish their value for such downstream uses. The *Semantic Similarity* metric therefore helps to ensure that the generated outputs span a broader range of plausible requirements within each domain.

Combined, the three metrics *Completeness*, *Degree of Realism*, and *Semantic Similarity* form an integrated evaluation framework, each targeting a distinct but complementary dimension of SSRS quality. This framework provides the foundation for assessing whether the generated artefacts fulfil their intended role as realistic, comparable, and diverse substitutes for real-world SyRS.

5.3.4. Rationale for Metric Selection

The evaluation metrics introduced in the preceding sections were selected to align directly with the overarching process requirements defined in this thesis: *R2-Comparability*, *R1-Realism*, and *R3-Diversity* as defined in section 4.1. Each metric operationalises one of these requirements in a manner that supports both the analysis and refinement of the SSRS generation process. Specifically, *Completeness* ensures that all generated SSRSs conform to a shared structural template, thereby enabling consistent evaluation and comparison. *Degree of Realism* addresses the plausibility and contextual fidelity of the generated specifications, which is essential for their intended role as stand-ins for real-world system requirement documents. *Semantic Similarity*, finally, is employed to assess the distinctiveness of SSRSs generated within the same domain, thereby supporting the criterion of diversity and preventing redundant outputs. Together, these three metrics form a coherent and task-specific evaluation framework tailored to the unique requirements of generating and assessing SSRS. No additional metrics were introduced, as these were found to sufficiently capture the essential quality dimensions necessary for the scope and objectives of this thesis.

6. The SSRS-Gen Process

In the context of this thesis, a structured generation process was conceptually designed to create Synthetic System Requirements Specifications (SSRSs) using an LLM. This chapter presents the SSRS-Gen process model on a conceptual level. To fulfil the defined requirements presented in chapter 4, the process was divided into distinct phases that collectively enable the generation of SSRSs, their automated evaluation along the dimensions of completeness, realism, and semantic diversity, and the iterative refinement of prompts based on evaluation results. Each phase is described in detail, including its objectives, rationale, and relation to the overall process structure. The chapter thereby provides a comprehensive overview of the logical architecture and reasoning behind SSRS-Gen. The subsequent Chapter 7 then presents the concrete implementation of this conceptual process using a real LLM system.

6.1. High-Level Overview

As illustrated in figure 6.1, the SSRS-Gen process is structured as a three-level iterative framework designed to meet the requirements defined in section 4.1. At the highest level, the entire process is executed for a pre-defined number of global iterations. This outer loop allows for prompt refinement and progressive improvement over time, addressing *R5–Zero-Resource Operation* and *R6–Expert Independence*. Since no high-quality SyRS documents or domain experts are available, early outputs are expected to be suboptimal and require iterative adjustment.

Each global iteration consists of two distinct sub-processes, which are visually represented in blue and green in the figure. The blue sub-process handles the core generation pipeline, while the green sub-process includes post-generation analysis and prompt refinement tasks. Within the blue sub-process, a middle iteration loop runs over a fixed number of m industry domains. This per-domain loop directly reflects *R4–Domain Independence* by ensuring that SSRS-Gen can produce outputs for a diverse set of industry domains. For each of these m domains, a third-level inner loop generates n SSRSs, with $n > 1$. This is necessary to support *R3–Diversity*, as multiple outputs are required within the same domain to assess and ensure semantic diversity.

The blue sub-process contains four sequential activities. Activity 1, *Generate SSRS*, is the core generative step and produces a single SSRS for a given industry domain using a structured prompt based on a pre-defined SSRS template. Activity 2, *Assess Completeness*, directly supports *R2–Comparability* by evaluating structural adherence to this template, ensuring consistent structure and content coverage across outputs. Activity 3, *Assess Degree of Realism*, addresses *R1–Realism* by evaluating whether the generated

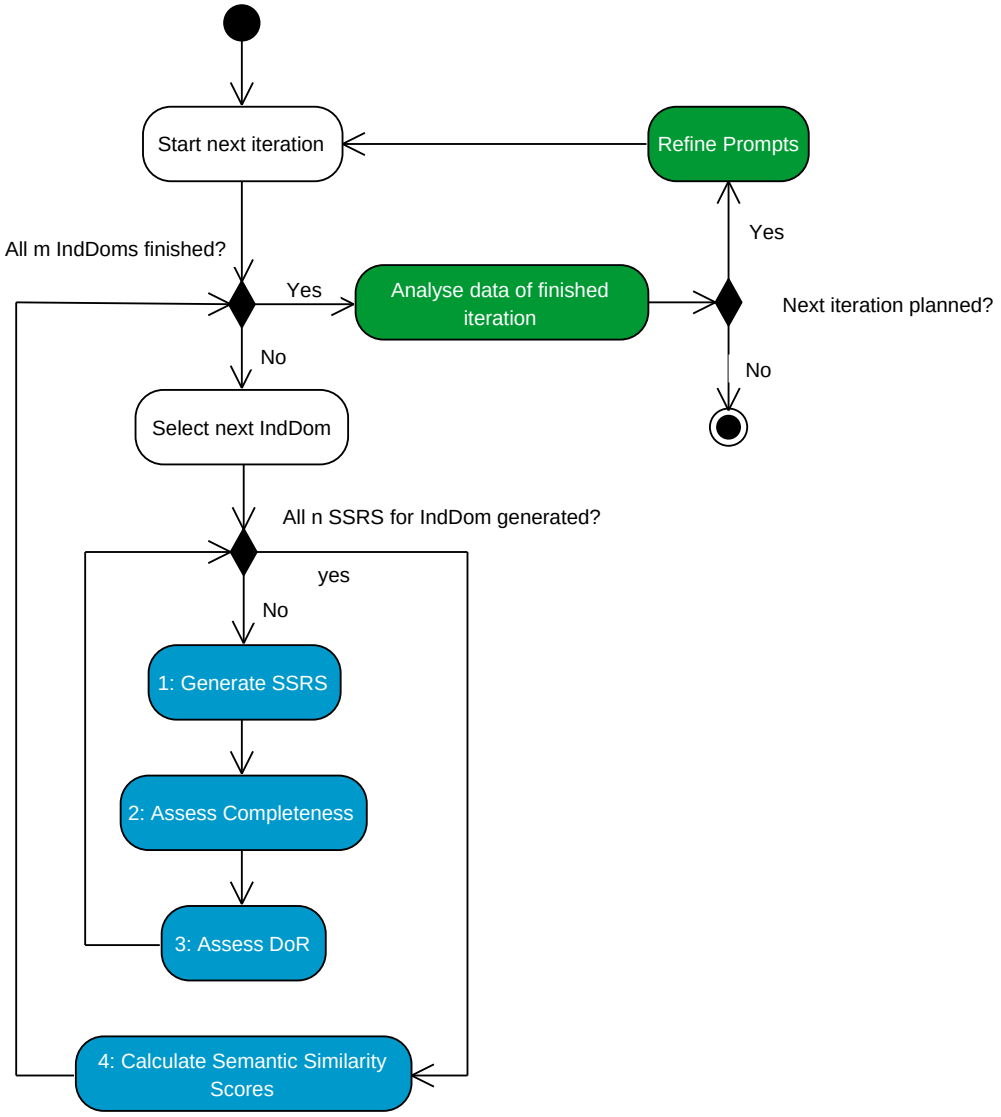

Figure 6.1.: High-level overview of the SSRS-Gen process.

SSRS plausibly resembles a real-world SyRS. Once n SSRSs have been generated for a given domain, Activity 4, *Calculate Semantic Similarity Scores*, is used to compute all $m \times \binom{n}{2}$ pairwise similarity values among the outputs, providing a quantitative basis for assessing semantic diversity.

After all blue activities are completed for all m domains, the process transitions to the green sub-process. This begins with *Analyse Data of Finished Iteration*, a manual review of the $m \times n$ SSRSs, their completeness and realism assessments, and the $m \times \binom{n}{2}$ similarity scores. Based on this analysis, *Refine Prompts* is performed to adjust and improve the prompts used in the next iteration. It is important to note, that within each global iteration, the same version of the SSRS generation prompt is used across all selected IndDoms. Prompt refinements are applied only between iterations, based on a combination of quantitative evaluation results and qualitative observations. This iterative feedback loop enables progressive improvement in output quality without relying on external training data or expert feedback. Over successive iterations, prompt refinements accumulate, gradually enhancing the generation process. This design choice is particularly important in the black-box setting of LLMs, where the model itself cannot be modified, and prompt engineering is the only lever available to influence output behaviour.

To provide a deeper understanding of the SSRS-Gen process, the following sections examine each of the four blue activities in detail. These include the SSRS generation, completeness assessment, DoR evaluation, and semantic similarity analysis.

6.2. Phase 1: SSRS Generation

```
You are a highly experienced Requirements Engineer.
Generate an SSRS based on the following template for the Industry
Domain <insert IndDom here>.

Template:
<insert template here>
```

Source Code 6.1: Basic example prompt for SSRS generation.

The first of the four sequential phases in SSRS-Gen focuses on the initial generation of SSRSs using a structured, zero-shot prompting approach. The objective of this phase is to generate SSRS for a given IndDom. As the core generative step of the SSRS-Gen process, this phase provides the foundation for all subsequent evaluation phases. SSRS generation requires three primary inputs: (1) a selected IndDom, (2) a predefined SSRS template that constrains the structure and types of requirements to be included, and (3) a zero-shot prompt used to initiate the generation process. A simplified instance of an SSRS generation prompt is displayed in listing 6.1 including placeholders for the selected IndDom and the predefined SSRS template as well as task instructions. The SSRS template, defined in section 3.4, specifies the expected structural format and requirement

categories to ensure consistency and comparability of SSRSs generated across different domains. The SSRS-Gen process does not impose strict limitations on IndDom selection, as the approach is intended to be domain-independent. This, however, assumes that the LLM used has been trained on sufficient data relevant to the chosen IndDom. Otherwise epistemic uncertainty effects can degrade SSRS quality, especially regarding their degree of realism.

All SSRSs are generated and evaluated without human intervention, ensuring that the results exclusively reflect the behaviour of the language model and the applied prompting strategy.

6.3. Phase 2: Completeness Assessment

Once an SSRS has been generated, the next step is to assess its structural integrity by verifying whether all required template elements are present. The objective of this phase is thus to evaluate whether a generated SSRS includes all elements specified in the predefined SSRS template. For instance, the template specifies sections such as *System Purpose* or *Stakeholders*, thus corresponding content must be present in the SSRS for it to be considered complete. Completeness is assessed exclusively with respect to the structural coverage of all required sections. The evaluation does not account for the quality of the generated section content. Assessment is conducted in binary terms: an SSRS is classified as *complete* if all required sections are present, and as *incomplete* if even a single required element is missing. This strict all-or-nothing criterion was adopted due to its alignment with the overarching goal of producing structurally *comparable* SSRSs. Incomplete outputs would undermine this comparability, thereby reducing the usefulness of the generated data for downstream evaluation or testing tasks.

All SSRSs proceed to the next phase regardless of their completeness classification. The results of these assessments are collected and analysed after the completion of each iteration and inform subsequent prompt refinements.

6.4. Phase 3: Degree of Realism Assessment

After verifying structural completeness, the next step is to evaluate the degree of realism. Specifically, whether a generated SSRS plausibly reflects a real-world SyRS within its respective IndDom. In the first iterations the DoR score was measured as a boolean value of *realistic* or *unrealistic*. However, during the iterative development of the SSRS-Gen process, the binary evaluation was found to be too shallow, as it offered little insight into the actual degree of realism. Consequently, it was replaced with a continuous value on a rational scale from 0 to 1 to enable a more nuanced assessment. No manual filtering or human intervention is applied.

All SSRSs proceed to the next phase regardless of their DoR classification. The results of the DoR assessment are collected and analysed after the completion of each iteration and inform subsequent prompt refinements.

6.5. Phase 4: Semantic Similarity Measurement

With structural completeness and realism assessed, the final phase evaluates the semantic similarity between SSRSs generated within the same IndDom. This phase is essential to ensure that the SSRS-Gen process produces a diverse set of SSRSs rather than redundant outputs, as semantic redundancy would diminish their value for downstream tasks such as evaluation and testing. The input to this phase consists of the three SSRSs generated for a single IndDom. Pairwise comparisons are conducted to quantify the semantic similarity between each SSRS pair. The resulting scores reflect the degree of semantic overlap and serve to identify whether the generated outputs are meaningfully distinct or redundant.

6.6. Iterative Refinement

Following the execution of all four core phases for each industry domain within a given iteration, the SSRS-Gen process proceeds with an iterative refinement step. As outlined in section 6.1, the purpose of this refinement loop is to improve output quality over multiple iterations, in alignment with the process requirements.

The refinement process is fully manual and centres on improving prompt design based on a systematic analysis of the collected evaluation results. These results include completeness and realism classifications, as well as semantic similarity scores. Prompt modifications are informed by three prioritized sources of feedback: (1) quantitative evaluation metrics and their variability, (2) qualitative issues surfaced during realism assessments, and (3) language-level concerns such as generic phrasing, reviewed by a non-expert human reader.

Refinements are implemented through the gradual introduction of scientifically validated prompt engineering techniques and prompt patterns from the list of selected patterns presented in chapter 5. Because the effects of individual patterns cannot be reliably predicted in advance, their prioritisation is based on an informed estimation of likely benefit, drawing from findings in the prompt engineering literature. Rather than exhaustively exploring all possible combinations of techniques and patterns, a pragmatic, iterative approach is adopted. Refinements are introduced sequentially and selectively, based on weaknesses identified in prior iterations. A comprehensive exploration of all possible combinations of prompt patterns and techniques is impractical due to the substantial variation involved, not only in the choice of patterns themselves, but also in their concrete formulation and application (e.g., the selection of different professional roles in the persona pattern). Given this complexity, the iterative refinement strategy represents a pragmatic and targeted approach. It enables the systematic improvement of prompt design based on observed weaknesses in the generated outputs, while maintaining methodological focus and feasibility.

7. Process Implementation

This chapter presents the technical implementation of the SSRS-Gen process outlined in chapter 6. While the previous chapter described the conceptual structure and rationale of the iterative generation, evaluation, and refinement framework, the current chapter details how this framework was operationalised using specific models, tools, and workflows. The implementation adheres to the exact same structure and iterative loops, as illustrated in figure 6.1. For each of the core generation and evaluation steps highlighted in blue, a dedicated section outlines the implementation details, describing how these activities were operationalised using an LLM and supporting tools.

In addition, this chapter includes supporting components essential for the overall execution of the SSRS-Gen process. These comprise the selection of tools and an LLM, the definition of a stopping criterion, and the setup of the iterative refinement loop based on data-driven prompt adjustments. Together, these components ensure transparency and reproducibility, and offer a comprehensive foundation for applying or adapting the SSRS-Gen process in comparable contexts.

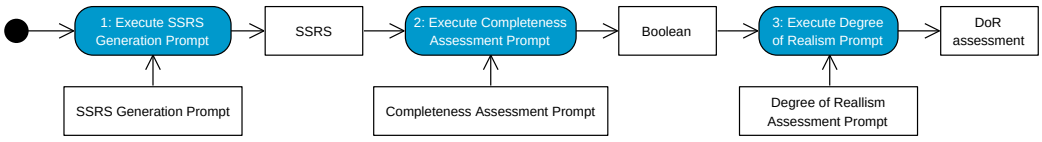

Figure 7.1.: Representation of the complete workflow for a single SSRS, showing sequential prompt execution for generation, completeness assessment, and DoR evaluation.

The activity diagram in figure 7.1 illustrates the sequential execution of prompts involved in processing a single SSRS. This detailed view corresponds directly to the first three blue activities of *Generate SSRS*, *Assess Completeness*, and *Assess Degree of Realism*, from the high-level process diagram in figure 6.1, outlining the internal structure of a single SSRS generation procedure. In this implementation, a single LLM is used not only for generating SSRSs, but also for evaluating them through a Self-Assessment approach. Specifically, the same model is prompted to assess both the completeness and DoR of the SSRS it generated. The workflow begins with the execution of the SSRS generation prompt, which produces an SSRS for the current IndDom. This output is then passed to the completeness assessment prompt, which verifies whether all required template elements are present and returns a binary result. Finally, the DoR assessment prompt is applied to evaluate the plausibility of the SSRS content, also yielding a boolean

outcome.

The fourth blue activity of *Calculate Semantic Similarity Scores* is not depicted in this diagram, as it operates on the set of all n SSRSs generated within a domain and requires pairwise comparisons. Unlike the first three activities, which process a single SSRS in sequence, this step is performed in batch after the full set of domain-specific outputs has been generated.

7.1. Tooling and Model Selection

The implementation of the proposed SSRS-Gen process relies on a combination of web-based tooling, local processing, and structured version control to ensure traceability and reproducibility. ChatGPT-4o is employed as the sole LLM for all stages of the process, including SSRS generation, completeness assessment, and DoR evaluation. All model interactions are conducted via the OpenAI web interface. To maintain isolation and traceability, a new conversation is initiated for each IndDom, and these conversations are organised by iteration using the "projects" feature of the interface. This setup allows all generated data to be reconstructed if needed. ChatGPT-4o was selected because it was the most recent ChatGPT model publicly available at the start of this thesis and is extensively researched in scientific literature. The web interface was chosen over the API due to budget constraints. All prompt versions used during the iterative process, covering generation, completeness assessment, and realism evaluation, are maintained in a version-controlled format. Generated SSRSs are stored in individual text files, and all corresponding evaluation outputs are stored in a structured JSON file. A GitLab repository is used to manage all project artefacts, including prompt versions, generated SSRSs, evaluation results, and analysis scripts, thereby supporting reproducibility and organised data management.

Semantic similarity scoring was performed in a local Python environment using the `sentence_transformers` library. The pre-trained `all-mpnet-base-v2` model was selected due to its high accuracy across a range of semantic textual similarity tasks, as documented in the library's official benchmark evaluations. Cosine similarity is used as the default metric to compute pairwise similarity scores between SSRS instances. The resulting decimal scores range from 0 to 1, where a value of 1 indicates complete semantic equivalence and 0 indicates no semantic similarity [RG19].

7.2. Stopping Criterion

The SSRS-Gen process was executed for a total of ten iterations. Each iteration produced 30 SSRSs, three per IndDom across ten selected IndDoms, resulting in a final dataset of 300 generated SSRSs. The decision to use ten IndDoms was motivated by the need to ensure sufficient contextual diversity without introducing overly niche or highly specialised domains. To compile this list, ChatGPT-4o was prompted to propose ten reasonably distinct and broadly applicable industry domains. The resulting list was manually reviewed to confirm that each domain was unique, widely relevant, and

understandable without further modification. This approach ensured a balanced set of IndDoms suitable for evaluating the generalisability of the SSRS-Gen process. Within each IndDom, three SSRSs were generated per iteration to enable pairwise semantic similarity comparisons. This design yields three unique comparisons per domain, providing an adequate basis for assessing semantic diversity while keeping the evaluation workload manageable. Since the evaluation involves open-ended, semantic data rather than categorical or numerical labels, it was necessary to strike a balance between analytical depth and human feasibility. A total of 300 SSRSs is considered a reasonable sample size for initial exploration, allowing for the identification of trends and process effects without exceeding the practical limits of prompt execution, data handling, and qualitative interpretation. Expanding the dataset further was not pursued, as it would have incurred disproportionately higher effort without a guaranteed increase in analytical value, particularly in the absence of a validated ground truth. The lack of such a ground truth also ruled out the application of conventional supervised evaluation metrics such as precision, recall, or F1-score. Constructing a reliable ground truth would have required extensive input from domain experts to generate, verify, and annotate a large set of high-quality SSRSs. Given the exploratory nature of this work and the complexity of the data, the expected benefit of creating such a reference did not justify the required effort.

7.3. Phase 1: SSRS Generation

Phase 1 constitutes the core generative step of the SSRS-Gen process, where individual SSRS documents are produced using a structured prompting approach grounded in a fixed template. The SSRS template, defining the expected structure and content of each SSRS, remains constant throughout the process, except for a single early revision to incorporate additional content. This updated version is used consistently in all subsequent iterations. The complete SSRS template, including detailed descriptions for each section, is documented in section 3.4. Prompt execution is performed manually using the OpenAI ChatGPT-4o web interface. For each IndDom, a temporary text file is prepared that contains the current versions of the SSRS generation, completeness assessment, and DoR prompts. Before execution, IndDom-specific placeholders within the prompt are manually replaced with the current IndDom name. The resulting IndDom-specific prompts were copied and pasted into the web interface three times in a row. This manual process supports efficient execution while adhering to OpenAI's usage policy[1] prohibiting automated use of ChatGPT's web interface.

All three SSRSs for a given IndDom, along with their corresponding completeness and DoR assessments, are generated within a single conversation. This design choice ensures that the model retains contextual memory across multiple outputs, allowing it to consider previously generated SSRSs when creating subsequent ones. This setup facilitates semantic diversity within each domain, as the model can be explicitly instructed to avoid

[1] `https://openai.com/policies/usage-policies/`

repetition or encourage variation based on prior responses. No post-processing or manual filtering is applied to the generated outputs. This non-intervention policy ensures that the results of the completeness and DoR evaluations reflect only the effectiveness of the prompt design and model behaviour, free from human correction or bias.

7.4. Phase 2: Completeness Evaluation

The completeness metric is assessed using an LLM self-assessment strategy. This decision is based on the nature of the task, which involves checking whether a generated SSRS contains all required sections defined in the SSRS template. Given that LLMs are well-suited for structured text analysis and pattern recognition, this task presents relatively low complexity and aligns with the model's inherent capabilities. Moreover, automating this evaluation significantly increases time efficiency, as the model can perform structural comparisons far faster than a human evaluator. The only inputs to this phase are the SSRS generated in Phase 1 and a dedicated prompt instructing the model to verify whether the output fully instantiates the SSRS template. Completeness assessment is executed immediately after the SSRS generation and within the same ChatGPT conversation for a given industry domain. The model is instructed to give a final assessment as a binary classification: `true` if all required sections are present, or `false` if any required element is missing. All evaluation results are recorded and subsequently used to inform prompt refinements at the end of each iteration. No manual intervention or correction is applied during this phase, ensuring that the recorded outcomes reflect the autonomous behaviour of the model under the given prompt configuration.

7.5. Phase 3: Realism Evaluation

Due to the limited availability of real SyRS data and feedback from IndDom experts, the *degree of realism* is assessed using an LLM self-assessment approach. A dedicated prompt instructs ChatGPT to evaluate the realism of its own previously generated SSRS. An overview of the research underpinning self-assessment approaches is provided in chapter 5. This evaluation strategy was selected because involving human IndDom experts throughout the iterative process would introduce significant delays and render the process impractical. More importantly, such reliance would contradict the primary objective of this thesis: to develop an SSRS LLM generation process suited for settings where access to expert knowledge and real SyRS data is inherently scarce. The DoR assessment is executed immediately after the completeness evaluation within the same ChatGPT conversation. A structured prompt is used to determine whether the SSRS is plausible, coherent, and representative of a realistic SyRS within the specified industry domain. The evaluation produces a binary result: `true` (realistic) or `false` (not realistic). As mentioned in the previous chapter, the boolean format was replaced with a continuous value on a rational scale from 0 to 1 after the first iterations, to enable a more nuanced assessment.

The results are stored in a structured JSON format for each SSRS. While the completeness and DoR assessments are designed to operate independently, both are conducted by the same LLM within a shared conversation context. As a result, implicit dependencies may arise due to the model's internal state or reasoning, despite no such dependencies being explicitly defined in the prompts. The implications of this setup are discussed further in chapter 11. To support prompt refinement, qualitative feedback generated by the LLM during the DoR assessment is reviewed and analysed by a human non-IndDom expert. This feedback informs adjustments not only to the SSRS generation prompt but also to the DoR evaluation prompt itself. As with all process phases, no manual filtering or correction is applied to the generated outputs during this step.

7.6. Phase 4: Similarity Scoring

Once three SSRSs are generated for a given IndDom, semantic similarity scoring is conducted independently of the LLM-based workflow. This phase is implemented locally in a Python environment using the `sentence_transformers` library, which has been validated in scientific literature for computing embedding-based semantic similarity between natural language texts [RG19]. The process is fully automated and does not require any manual review or intervention.

Semantic similarity is assessed for each of the three SSRS pairs within the domain. For each pair, a continuous score in the range [0, 1] is computed, where 1.0 indicates high semantic equivalence and 0.0 denotes complete dissimilarity. The resulting scores, along with metadata identifying the corresponding SSRS pairs, are stored in the structured JSON file associated with the respective IndDom. In addition, the average of the three pairwise scores is calculated and recorded as a summary metric representing overall semantic diversity within the domain. To contextualise the interpretation of similarity scores, a baseline analysis was conducted in which ten SSRSs, one from each IndDom, were compared to an "empty" SSRS template. This template included only the section headings without any explanatory or content-specific text. The resulting similarity scores averaged at approximately 0.45, indicating that structural overlap of shared headings alone introduces a non-negligible baseline similarity. This implies that semantic similarity scores generated during the iterative process should not be interpreted relative to a theoretical minimum of 0.0, but rather to this empirical lower bound of 0.45. However, this structural baseline does not account for domain-specific similarities that are likely to occur naturally within a given IndDom. For instance, in the Healthcare domain, many SSRSs will inevitably reference recurring entities such as "patients" as stakeholders or refer to common compliance requirements. Such domain-inherent commonalities further elevate the minimum achievable similarity score, even in otherwise diverse outputs. Consequently, the true lower bound for semantic similarity within a single domain is likely to lie above 0.454. This re-framing ensures a more accurate and context-sensitive interpretation of semantic diversity in the generated SSRS sets.

Similarity scores are not used to exclude or filter SSRSs during iteration. All outputs are retained regardless of similarity level. As with completeness and realism evaluations,

similarity results are analysed post-iteration to inform subsequent prompt refinements.

7.7. Data Analysis and Prompt Refinement Loop

Upon completion of each full iteration of SSRS generation and evaluation across all ten IndDoms, the resulting data is systematically analysed to assess the effectiveness of the current prompt configurations. For each of the three evaluation metrics, completeness, degree of realism, and semantic similarity, average scores are computed across all domains. Additionally, domain-specific metric scores and their variability are examined to identify inconsistencies, outliers, or domain-dependent patterns. To facilitate this analysis, a custom Python script is employed to visualise metric trends across current and previous iterations, enabling comparative evaluation and clearer insight into the effects of prior prompt modifications. Prompt refinement is informed by a layered feedback framework. First, quantitative signals such as low DoR scores, high semantic similarity scores or highly fluctuating metric scores are examined. Second, qualitative insights are drawn from the model's feedback during DoR assessments, which can reveal important issues in content plausibility or specificity. Third, human judgment is incorporated through targeted review by a non-IndDom expert, with particular attention paid to superficial or generic phrasing (e.g., vague statements such as "the system should be reliable" that lack concrete, domain-specific detail). Notably, changes in evaluation metrics are not interpreted purely numerically: a worsening of an evaluation metric, such as a lower average DoR score, may reflect increased scrutiny or higher assessment granularity rather than a true decline in realism. For example, a more refined DoR prompt may highlight issues previously overlooked, thereby lowering scores while simultaneously improving diagnostic value. Accordingly, both positive and negative changes in metric values are interpreted in the broader context of output quality, informativeness, and their utility for guiding further refinement.

Refinements target not only the SSRS generation prompt but also the evaluation prompts for completeness and DoR. Particular attention is given to the DoR assessment prompt, which is considered equally critical in the refinement loop. Inaccurate realism evaluations risk distorting the feedback cycle and may result in ineffective or counterproductive prompt adjustments. To support traceability, all prompt versions are stored independently, and for each iteration, the exact combination of prompt versions used is recorded. Based on the insights derived from metric trends, model-generated feedback, and human review, targeted prompt modifications are implemented before starting the next iteration. These refinements may involve the introduction of new prompt engineering techniques or patterns from the pre-defined set of patterns defined in chapter 5, or the rewording and extension of existing prompt instructions. Special emphasis is placed on the precise formulation of prompt pattern and technique implementations, as the effectiveness of a prompt pattern or technique is not inherent to its presence alone, but depends critically on how it is articulated. Therefore, each refinement cycle includes a focused effort to improve prompt phrasing, ensuring clearer task descriptions and minimising ambiguity regarding both the desired and undesired model behaviours.

8. Process Execution Results

This chapter presents the results of ten iterative runs of the SSRS generation process as presented in chapter 7, including evaluations of completeness, degree of realism, and semantic similarity. Each iteration introduced prompt refinements aimed at improving output quality, ranging from revised SSRS generation instructions to updates in the evaluation prompts. The metrics, completeness, degree of realism, and semantic similarity, are reported across all iterations to illustrate how successive prompt adjustments influenced the quality of the generated SSRSs. Table 8.2.2 documents the evolution of the SSRS generation prompt, from the initial version (v1.0) containing basic instructions to the final version (v1.9), which incorporates more advanced constraints and structural guidance. Tables 8.2.3 and 8.2.4 provide corresponding version histories for the completeness and realism evaluation prompts. The following sections present and describe the metric trends observed over the course of the iterations, with a focus on how these correlate with prompt updates.

To support the interpretability of the reported metric outcomes, Figures 8.1 to 8.4 illustrate a complete SSRS generated in the final iteration for the finance industry domain. The example demonstrates full adherence to the SSRS template structure, with each section and sub-element populated by contextually appropriate content. ChatGPT-generated responses are formatted as concise, domain-specific bullet points, reflecting the model's ability to instantiate structured requirements in alignment with the provided SSRS template. This example serves as a concrete reference for illustrating the structural format and level of detail produced in the final iteration of the process. It serves as a contextual background for interpreting the structural trends reported in the following sections.

To preserve the readability of this section, large and detailed charts are displayed in the Appendix A instead.

1. System Overview

System Purpose:
The system, named "TradeFlow," is designed to enable seamless management of cross-border trade financing for financial institutions and corporate clients. Its primary objective is to streamline the approval, tracking, and settlement of trade finance products (e.g., letters of credit, guarantees, supply chain financing) while ensuring compliance with international trade regulations.

Domain/Context:
The system operates within the trade finance sector of the financial industry, catering to financial institutions, multinational corporations, and SMEs engaged in global trade.

Stakeholders:

- **Corporate Clients:** Importers/exporters requiring trade financing.
- **Relationship Managers:** Bank employees managing corporate client portfolios.
- **Compliance Officers:** Overseeing trade compliance and fraud prevention.
- **Back Office Staff:** Responsible for processing and approving trade finance transactions.
- **Auditors:** Internal and external teams reviewing transaction logs and compliance reports.

User Base Characteristics:

- **Size:** Medium-sized user base (~10,000 users initially, scaling to ~50,000).
- **Diversity:** Users include multinational corporations, SMEs, bank employees, and third-party auditors.
- **Geographic Distribution:** Users span multiple regions, with concentration in Asia, Europe, and North America.
- **Roles:** Clients (request trade finance), internal bank staff (process approvals), auditors (review logs).

Operational Environment:

- **Hosting:** Hybrid cloud environment to support scalability and data sovereignty needs.
- **Access:** Web-based platform for corporate users, desktop applications for internal staff, and mobile access for relationship managers.
- **High Availability:** 24/7 operations with support for time-zone differences across continents.

Usage Scenarios:

1. A corporate client applies for a letter of credit for a large import.
2. A relationship manager reviews the application, verifies trade details, and forwards it for approval.
3. Compliance officers perform checks for anti-money laundering (AML) and sanctions screening.
4. Back office staff finalize approval and issue the letter of credit to the counterparty bank.
5. Auditors periodically review logs for regulatory compliance and anomalies.

Figure 8.1.: System Overview section of an SSRS generated in the final iteration for the finance industry domain.

2. Functional Requirements

Core Features:

- **Trade Finance Application Management:** Submission, tracking, and approval workflows for trade finance products.
- **Compliance Integration:** Automated checks for AML, KYC, and sanctions compliance.
- **Document Management:** Secure upload, storage, and validation of trade documents.
- **Settlement Processing:** Automated disbursement and reconciliation of trade finance payments.
- **Custom Reporting:** Configurable reports on transaction histories, pending approvals, and regulatory compliance.

Authentication Conditions & Frequency:

- Users authenticate once per session.
- Re-authentication required for sensitive actions, such as approving transactions above a threshold amount or accessing detailed compliance reports.
- Sessions expire after 30 minutes of inactivity.

Sensitivity of Actions & Permission Levels:

- **High Sensitivity:** Approving transactions, accessing compliance reports (Relationship Managers, Compliance Officers).
- **Medium Sensitivity:** Creating trade finance requests (Corporate Clients).
- **Low Sensitivity:** Viewing transaction history or general reports (General Users).
- Permissions are role-based and strictly enforced.

Figure 8.2.: Functional requirements section of an SSRS generated in the final iteration for the finance industry domain.

3. Non-Functional Requirements

Performance:

- Average response time: ≤ 2 seconds for standard operations.
- System throughput: 1,000 simultaneous transactions processed per second.

Scalability:

- Horizontal scaling to support up to 50,000 concurrent users and doubling transaction volume over five years.

Reliability:

- 99.95% uptime, with automated failover mechanisms in place.
- Recovery time objective (RTO): 15 minutes.
- Recovery point objective (RPO): 5 minutes.

Security:

- Role-based access control (RBAC).
- Encryption of data at rest and in transit.
- Periodic penetration testing and continuous vulnerability monitoring.

Usability:

- Intuitive user interface with role-specific dashboards.
- Multi-language support (English, Mandarin, Spanish, etc.).
- Mobile-responsive design for relationship managers.

Audit & Monitoring:

- Detailed logging of user actions, authentication events, and sensitive transactions.
- Automated alerts for unusual patterns, such as rapid sequential approvals.
- Regularly generated compliance reports for regulatory authorities.

Figure 8.3.: Non-functional requirements section of an SSRS generated in the final iteration for the finance industry domain.

4. Constraints

Technical Constraints:

- Must use existing trade finance APIs for integration with SWIFT and other global banking networks.
- Operates within a hybrid cloud infrastructure with specific components mandated to remain on-premise for compliance.

Compliance Requirements:

- Adherence to AML, Basel III, and FATF guidelines.
- GDPR compliance for user data originating from the EU.
- Local data residency requirements for transactions in countries like India and China.

Resource Constraints:

- Budget: $5 million for initial development and deployment.
- Timeline: 18 months, including development, testing, and deployment.
- Team Size: 50 members (developers, testers, business analysts, compliance experts, project managers).

Integration Needs:

- Integration with existing ERP systems used by corporate clients.
- Connectivity to SWIFT and other global trade networks for secure communication.
- Interfacing with internal bank systems for payment processing and customer records.

Figure 8.4.: Constraints section of an SSRS generated in the final iteration for the finance industry domain.

8.1. Completeness Results

Completeness is evaluated as a binary metric (`true`/`false`) by ChatGPT to determine whether all required sections specified in the SSRS template are present in a generated SSRS. During the iterative process, two minor adjustments were made to the completeness assessment prompt, both aimed at reducing unnecessary verbosity in the model's output. Specifically, changes focused on discouraging the model from repeating the entire SSRS content point-by-point, as this was not required to support the binary classification task. As mentioned in section 7.2, a total of 300 SSRSs were generated. From the first iteration onward, all completeness evaluations yielded a result of `true`, so none of the 300 completeness assessments resulted in an `incomplete` classification. Given the complete consistency of this result, no further quantitative analysis of this metric is provided in the subsequent sections. The implications and possible limitations of this outcome are addressed in detail in chapter 11.

8.2. Prompt Refinements

Throughout the iterative execution of the SSRS generation process, prompts were refined in response to observed limitations in output structure, realism, and diversity. Rather than applying all prompt engineering techniques at once, changes were introduced incrementally, enabling the identification of their individual effects. This section summarizes the most relevant refinements made to each prompt as well as the SSRS template.

8.2.1. SSRS Template

Version	Iteration	Key Changes / Notes
v1.0	1-4	Initial version of the SSRS template.
v1.1	5-10	Added new points to the template: "Operational Environment", "Usage Scenarios" and more authentication related details.

Table 8.1.: Changes in the SSRS Template.

The changes made to the SSRS template are displayed in table 8.2.1. It remained unchanged during the first four iterations, with version v1.0 defining the initial structure used to guide generation. In iteration five, version v1.1 was introduced, expanding the template to include the new elements of *Operational Environment*, *Usage Scenarios*, and more granular authentication-related requirements. These additions were not informed by deficiencies observed in earlier outputs, but were introduced to better align the SSRS content with the evaluation scope of the SecuRe constraint-based recommender system. The updated template was used consistently in all subsequent iterations to ensure comparability.

8.2.2. SSRS Generation Prompt

Version	Iteration	Key Changes / Notes
v1.0	1	Initial basic prompt, no prompt patterns or prompt engineering techniques
v1.1	2	Added the textual cue "Ensure that the scenario is distinct."
v1.2	3	Introduced an "expert persona" of "a highly experienced Requirements Engineer and Business Analyst specializing in <insert sector here> software systems." and clarified the textual cue to "Ensure that the scenario is distinct from any previously generated scenarios."
v1.3	4	Improved phrasing and emphasis on prioritising realism over enforcing diversity.
v1.4	5	Added chain-of-thought reasoning by prompting to go through "each section of the template step by step."
v1.5	6	Added instruction to exclude explicit references to authentication methods; added a self-assessment for contradictory requirements.
v1.6	7	Changed self-assessment instructions to focus solely on matching allocated resources with the requirements and constraints.
v1.7	8	Expanded and clarified instructions for SSRS diversity.
v1.8	9	Removed the overly narrow and specific focus of the self-assessment on allocated resources.
v1.9	10	Finalized version based on analysis of effects of prompt changes in all previous versions.

Table 8.2.: Prompt refinements for the SSRS Generation prompt.

The SSRS generation prompt was refined iteratively across ten versions, with each change addressing observed issues or adding prompting techniques from the pre-defined list to improve output quality. Version v1.0 served as a baseline, containing only a minimal instruction to generate an SSRS based on a provided template for a specified industry domain: "Generate a scenario, in plain-text format, based on the following template for the sector <insert sector here>. Template: <insert template here>". In v1.1, a basic cue was added to encourage diversity in outputs. This was strengthened in v1.2 by introducing the persona pattern, prompting ChatGPT to answer from the viewpoint of "a highly experienced Requirements Engineer and Business Analyst specializing in <insert sector here> software systems". Additionally the diversity instructions were clarified, prompting ChatGPT to explicitly distinguish new SSRSs from previously generated ones. In v1.3, the emphasis shifted toward realism by de-emphasizing forced diversity when it conflicted with plausibility. Version v1.4 introduced a chain-of-thought prompting approach by instructing the model to proceed step by step through

the SSRS template, thereby promoting more thorough and structured content generation. Version v1.5 introduced two notable changes: a restriction prohibiting references to specific authentication methods, and a self-assessment component to detect contradictory requirements. The restriction was added because generated SSRS are intended to be used for evaluating the SecuRe recommender system in subsequent projects. In v1.6, the self-assessment component was revised to focus exclusively on evaluating the consistency between allocated resources and the declared requirements. This refinement was motivated by repeated observations from the DoR self-assessments, which frequently highlighted discrepancies between stated resource availability and the feasibility of the specified requirements. In v1.7, the diversity instructions were revised for greater clarity and specificity. Version v1.8 subsequently removed the narrow self-assessment related to resource adequacy due to its limited impact on output quality. The final version, v1.9, integrated and consolidated all effective changes based on an analysis of the previous iterations.

Figure A.1 presents the final version of the SSRS generation prompt employed in the tenth iteration. The prompt is composed of several structured components designed to guide ChatGPT effectively through the SSRS generation process. It begins with the application of the persona pattern, assigning the model the role of a highly experienced Requirements Engineer and Business Analyst specializing in the target industry domain. This is followed by a main task description, instructing ChatGPT to generate a highly realistic and unique SSRS tailored to the challenges and goals of the specified IndDom. The core instructions are organized into clearly defined bullet points. First, the model is instructed to use the predefined SSRS template and ensure that all required sections are present, thereby supporting structural completeness. This is followed by an explicit constraint prohibiting references to authentication methods, introduced due to the intended application of the generated SSRSs in evaluating the SecuRe recommender system. Instructions regarding SSRS diversity, which in earlier iterations were framed in vague terms such as “Ensure that the scenario is distinct,” were revised to include more concrete and interpretable guidance. In the final prompt version, the model is directed to generate SSRSs that differ from previous outputs, explicitly encouraging variation in system purposes and core features. A similar approach was taken to clarify the concept of realism. General instructions such as “Generate a realistic scenario” were replaced with a more specific definition of realism, emphasizing alignment with best practices and typical requirements relevant to the target IndDom. The prompt also incorporates a self-refinement mechanism combined with chain-of-thought prompting. This mechanism consists of two sequential sub-tasks: (1) verifying the coherence between allocated resources and the stated requirements and constraints, and (2) confirming that the newly generated SSRS is sufficiently distinct from previously generated SSRSs within the same conversation. ChatGPT is explicitly instructed to evaluate the drafted SSRS against these criteria and revise it accordingly if deficiencies are identified. The final instruction specifies that only the refined SSRS should be output, with all intermediate reasoning and evaluation steps omitted.

8.2.3. Completeness Evaluation

Version	Iteration	Key Changes / Notes
v1.0	1	Initial basic prompt for a boolean assessment plus justification.
v1.1	2	Clarified instructions to give a justification only for missing points.
v1.2	3 - 10	Reduced output verbosity and defined a structured output format.

Table 8.3.: Prompt refinements for the SSRS Completeness assessment prompt.

The initial version of the completeness assessment prompt (v1.0) served as a minimal baseline, containing only the essential task instructions. The model was instructed to verify whether the generated SSRS fully implemented the predefined template and return a boolean result accordingly. Additionally, it was asked to provide a justification to gain insight into its reasoning, particularly for possible cases where the SSRS is deemed incomplete. In version v1.1, the instructions were revised to address issues with excessive verbosity. Specifically, ChatGPT frequently responded by paraphrasing or restating the entire SSRS to prove completeness. To mitigate this, the prompt was modified to request justifications only when specific template elements were missing. However, the verbosity issue persisted, with the model continuing to produce unnecessarily long outputs that repeated much of the SSRS content. To improve clarity and enforce consistency, version v1.2 introduced a structured output format. This format required the model to list each template element and sub-element alongside a corresponding "`<True|False>`" label, followed by an overall completeness verdict. Justifications were only required for elements marked as missing. This revision substantially enhanced the readability, consistency, and interpretability of the model's assessments. The final version of the prompt, used from iteration 3 onward, is shown for reference in Figure A.2.

8.2.4. Degree of Realism Assessment

The initial version (v1.0) of the DoR assessment prompt served as a simple baseline without the application of any prompt patterns or techniques. It instructed ChatGPT to evaluate whether a given SSRS was realistic for its respective industry domain by responding with a binary true/false value, followed by a justification. No explicit criteria were defined for what constitutes realism, nor were any structured evaluation procedures included at this stage.

After the first two iterations the scale was changed from binary to decimal in version v1.1. The switch to a decimal DoR score between 0 and 1 was motivated by the limitations of true/false judgments in capturing partial realism. A binary assessment enforces an implicit threshold: SSRSs with minor but non-negligible flaws may still be marked as "realistic," suppressing valuable diagnostic information. By contrast, a decimal score reflects realism as a spectrum, allowing the model to account for varying degrees of

plausibility and internal coherence. This shift enables more granular feedback, supports better comparability, and facilitates targeted refinements in subsequent iterations.

Version	Iteration	Key Changes / Notes
v1.0	1 - 2	Basic prompt for a true/false realism evaluation with justification.
v1.1	3	Changed to a decimal realism score (0-1). 0 unrealistic, 1 realistic.
v1.2	4	Added persona: "Senior Requirements Engineer and Business Analyst with broad experience across various industries, specializing in the evaluation and analysis of business and technical requirements".
v1.3	5	Persona refined to domain expert: "[...] with extensive experience in <insert domain here> [...]"
v1.4	6	Extended instructions to deduct score points for each identified realism issue and not just provide an overall score.
v1.5	7	Added focus on resource alignment (time, budget, team) with SSRS requirements. Prompt the LLM to focus just on information within the confines of the template and nothing "missing" that is outside the template.
v1.6	8	Emphasized deductions to be proportional to the identified realism issue and to give a reasoning for the amount of deduction.
v1.7	9	Required step-by-step checking of all template points/subpoints and to use 0.01 increments for score deductions.
v1.8	10	Finalized version based on analysis of effects of prompt changes in all previous versions.

Table 8.4.: Prompt refinements for the SSRS Degree of Realism assessment prompt.

In version v1.2, a professional persona was introduced to improve the quality and depth of the realism evaluations. The prompt now instructed ChatGPT to act as a "Senior Requirements Engineer and Business Analyst with broad experience across various industries, specializing in the evaluation and analysis of business and technical requirements." This addition aimed to provide stronger domain-context awareness.

In version v1.3, the persona introduced in the previous iteration was refined to represent a domain-specific expert, described as having "extensive experience in <insert domain here>." This adjustment was made to better ground the realism evaluation in the specific context of each industry domain. By aligning the evaluator persona more closely with the IndDom of the SSRS under assessment, the goal was to improve the contextual relevance and accuracy of the evaluations.

In version v1.4, the prompt was modified to instruct the LLM to explicitly deduct score points for each identified realism issue, rather than providing only a single overall score. This change was introduced to encourage more detailed and structured feedback. It also

enabled a clearer distinction between minor and major realism violations, supporting a more nuanced interpretation of the realism score and helping to prioritize refinement efforts based on severity.

In version v1.5, two major adjustments were made. First, an explicit instruction was added to evaluate only the information provided within the SSRS, not to penalize missing information outside the defined template. This change was necessary because earlier evaluations often cited the absence of specific details as flaws, which distorted the realism score. The goal of the DoR self-assessment is to evaluate only the realism of the information provided, not its completeness, as the scope of included information is limited by the SSRS template. Second, the prompt was refined to increase focus on resource alignment, instructing the model to assess whether budget, timeline, and team size were appropriate for the functional and non-functional requirements. This emphasis was motivated by recurring feedback from earlier assessments that frequently identified unrealistic resource allocations.

In version v1.6, the prompt was revised to emphasize that score deductions should be proportional to the severity of the identified realism issues. This change was introduced in response to inconsistent and seemingly disproportionate deductions observed in earlier iterations. For instance, similar types of issues in different SSRS led to different score penalties. To mitigate this, the prompt was updated to require not only a list of issues but also a justification for the magnitude of each deduction. This was intended to guide the LLM toward more deliberate, calibrated scoring and to promote internal consistency across evaluations.

Version v1.7 introduced two key refinements to the DoR assessment prompt. First, the LLM was explicitly instructed to assess the SSRS step by step, covering all main template sections and their respective elements. This addition was motivated by the observation that previous assessments tended to focus disproportionately on a few sections, while others were rarely or never mentioned. This could of course just be due to the sections being realistic and thus not mentioned, but nevertheless it should be ensured that ChatGPT actually assesses each individual section of the generated SSRS. Second, the prompt mandated the use of 0.01 point increments for score deductions. This was implemented to overcome the coarse granularity observed in prior iterations, where ChatGPT often defaulted to 0.05 steps. The goal was to enable more fine-grained, precise evaluations and ensure that minor and major issues could be more accurately distinguished in the scoring.

In version v1.8, a finalized prompt was compiled by systematically analysing the effects of all prior prompt changes across the first nine iterations. The final version incorporated the most effective elements identified in earlier versions, aiming to maximize evaluation consistency, scoring precision, and IndDom-relevant feedback quality.

The final Degree of Realism (DoR) assessment prompt, shown in Figure A.3, incorporates several established prompt engineering techniques to support a structured and reliable evaluation process. It begins with the application of the Persona Pattern, assigning ChatGPT the role of a domain-specific expert with extensive experience in the corresponding industry. This persona is intended to enhance contextual sensitivity and

evaluation relevance. The task is clearly scoped to focus exclusively on the content of the SSRS, explicitly instructing the model not to penalize information that is absent but outside the defined template. The prompt continues with a detailed set of evaluation criteria, including instructions to assess each section and sub-point of the SSRS individually. A proportional scoring method is defined, requiring the model to deduct points in 0.01 increments based on the severity of identified realism issues. For each deduction, the model is instructed to provide a corresponding justification, promoting transparency and interpretability. The prompt concludes with an explicit output specification: the response must include a structured summary of identified issues, rationales for deductions, and a final aggregated realism score for the entire SSRS. This format aims to ensure scoring consistency, facilitate traceability, and support iterative prompt refinement.

8.3. Iteration-Level Insights

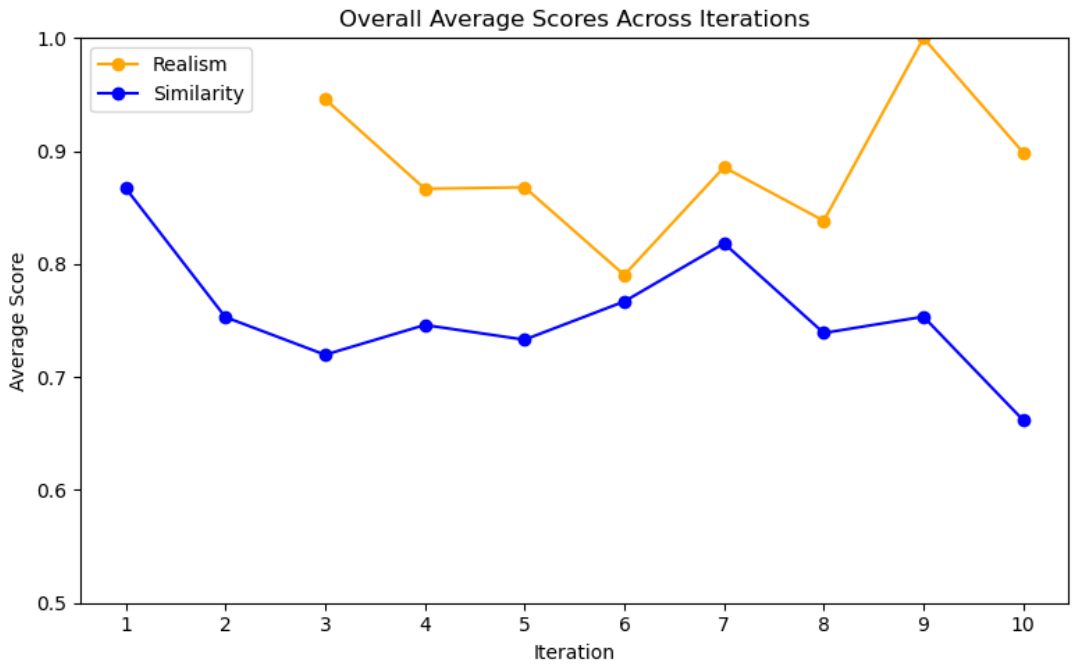

Figure 8.5.: The semantic similarity and degree of realism scores averaged over all 30 SSRS for a given iteration.

This section presents aggregated trends in DoR and semantic similarity scores across all ten iterations of the SSRS generation process. Each iteration consists of 30 SSRSs, three per IndDom, serving as the basis for computing iteration-level averages and score distributions for the two metrics. Figure 8.5 visualizes the evolution of DoR and seman-

tic similarity scores over the course of the process. It is important to note that high similarity scores indicate low semantic diversity among SSRSs within the same IndDom and are therefore considered undesirable. In contrast, lower similarity scores suggest greater variation and are aligned with the objective of generating diverse outputs. The subsequent sections provide a detailed examination of how SSRS quality, as quantified by DoR and semantic similarity, developed across iterations in response to prompt refinements.

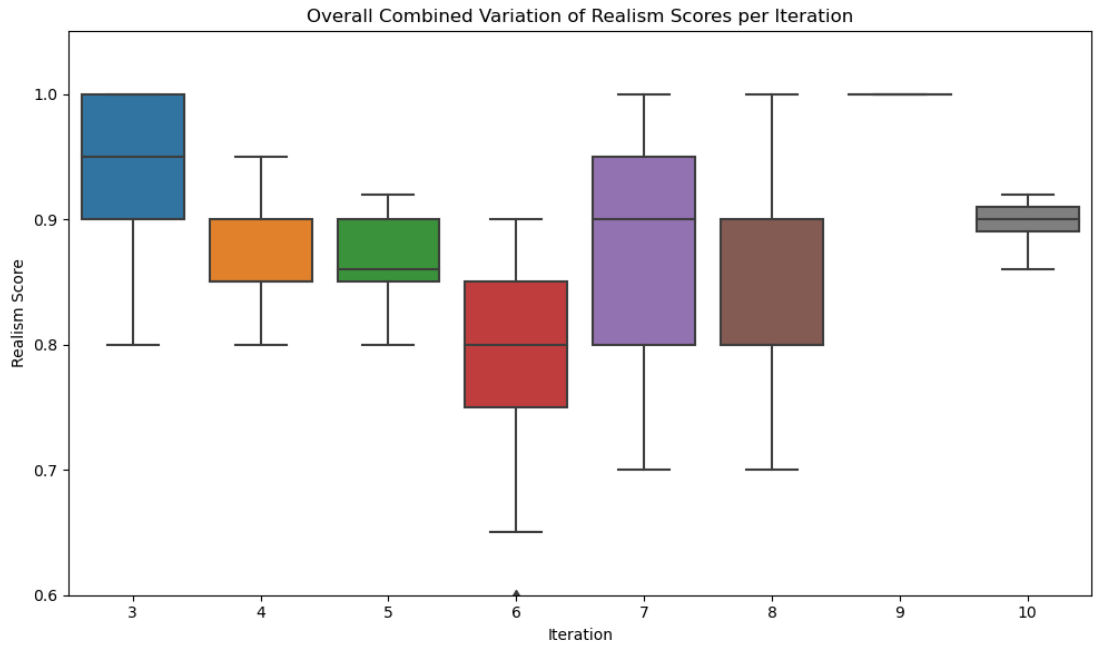

Figure 8.6.: Variation in DoR Scores aggregated across all 30 SSRS for each iteration.

8.3.1. Early Phase (Iterations 1 to 3)

In the first iteration, only similarity scores were assessed numerically; realism evaluation was conducted using a binary scale (true/false) and is therefore not represented in the aggregate metrics until iteration 3 as shown in figure 8.5. Besides the absolute score values it is also important to look at the variability of both metrics displayed in figure 8.6 and 8.7. For score variability small boxplots are desired as they display little variation, which indicates more consistent outputs by the SSRS generation process. With this context established, the following paragraphs examine the evolution of both metrics across iterations. The initial similarity average was relatively high, at 0.86, but decreased markedly to 0.71 by iteration 3. The second iteration showed the highest variability across the entire process, with similarity scores ranging from nearly 0.95 down to approximately 0.4. Iteration 3 introduced a decimal score for DoR, with an average of approximately 0.95, but again accompanied by substantial variation, with scores ranging from 0.8 to

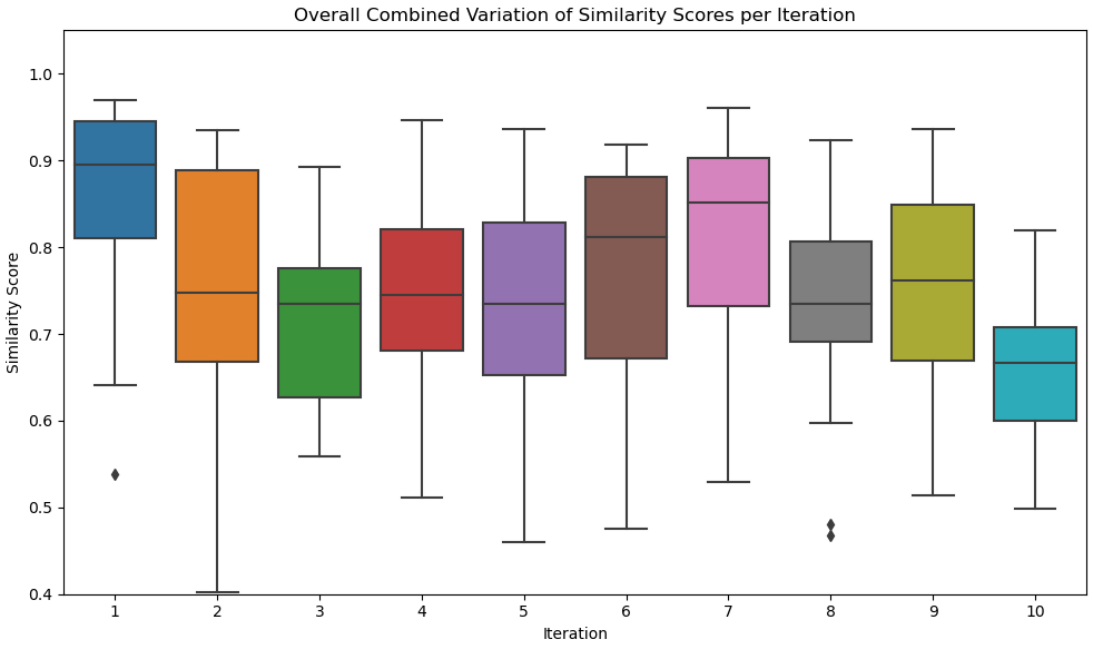

Figure 8.7.: Variation in Similarity Scores aggregated across all 30 SSRS for each iteration.

1.0. IndDom-specific patterns from the similarity heatmap, displayed in figure A.5 show that in the first iteration, IndDoms such as Education, Finance, Manufacturing, and Retail and Supply Chain scored particularly high, with values exceeding 0.9, reaching up to 0.96. E-commerce, by contrast, scored notably lower at 0.62. In iteration 2, most IndDoms experienced substantial score drops, for example Finance fell from 0.95 to 0.68, and Media and Entertainment dropped from 0.88 to 0.6. This shift occurred following a minimal prompt modification: the addition of the sentence "Ensure that the scenario is distinct."

In iteration 3, average similarity scores improved again down to a local minimum of 0.72, only surpassed by the last iteration thus displaying the second best average similarity scores of all iterations. Domain-level scores ranged from a low of 0.57 in Government and Public Services to a high of 0.84 in Logistics. No IndDoms exceeded a score of 0.9 in this iteration. The range remained high, with a 0.27 spread between the lowest and highest values. In this iteration, the SSRS generation prompt was extended with the Expert Persona pattern, instructing the LLM to adopt the role of a "highly experienced Requirements Engineer and Business Analyst."

The iteration subplots of similarity and realism scores, displayed in figure A.7, reveal that iteration 1 displayed fairly consistent similarity scores across most IndDoms, except for E-commerce. Iteration 2 introduced significant fluctuation in similarity across IndDoms. In iteration 3, realism scores began to appear, ranging from 0.85 to 1.0. In this

same iteration, similarity scores were generally lower than realism scores, with Healthcare being a notable exception. The magnitude of the gap between similarity and realism also varied considerably, with IndDoms such as Finance showing nearly a 0.4 point gap, while others such as E-commerce and Logistics had much smaller disparities.

The Boxplots for realism displayed in figure A.8, confirm that iteration 3 started with mixed levels of variability in realism: some IndDoms displayed near-zero variation, while others had significantly wider ranges. Similarity boxplots showed narrow score ranges for five to six IndDoms in iteration 1, while 3–4 IndDoms already exhibited broader variability. Variability increased in iterations 2 and 3, with higher ranges in most IndDoms.

8.3.2. Mid Phase (Iterations 4 to 7)

In the mid-phase, a general drop in realism averages was observed. Iterations 4 and 5 showed reduced average realism scores between 0.85 and 0.9, coupled with declining variability. However, iteration 6 marked a significant turning point, with realism dropping to a process-wide low of 0.79. This iteration also saw a major increase in score variability, with values ranging from 0.65 to 0.9. This high level of fluctuation persisted into iterations 7 and 8, although average scores were slightly higher than in iteration 6. Similarity trends during this phase show a gradual increase in average scores, peaking at a local high of 0.81 in iteration 7. Variability in similarity scores remained largely stable throughout iterations 4 to 7. The similarity heatmap in figure A.5 reveals that in iterations 4 and 5, domain-level scores continued to range broadly from 0.57 for Finance in iteration 4 to 0.93 for Healthcare in iteration 5. In iteration 5, the Chain-of-Thought pattern was introduced in the prompt, leading to a slight average score improvement, but also contributing to increased variability.

In iterations 6 and 7, average similarity scores declined again, positioning them among the lowest across all iterations except for the first. For instance, in iteration 6, Finance achieved a score of 0.55, the best individual similarity score overall, though this came with a wide internal range from approximately 0.48 to 0.69. Prompt modifications in these iterations aimed to improve realism, but appeared to negatively impact similarity in some IndDoms. Subplot analysis with figure A.7 shows that realism scores remained fairly stable across IndDoms in iterations 4 and 5. Similarity scores, however, continued to vary significantly. Notably, similarity exceeded realism scores in one IndDom in iteration 4, two IndDoms in iteration 5, five IndDoms in iteration 6, and four in iteration 7. Realism score stability diminished in iteration 6 and even more so in iteration 7 which displays highly fluctuating scores. Realism boxplots, shown in A.8, for iterations 4 and 5 display uniform variation and small score ranges, consistently below 0.05. By iteration 6, this consistency had eroded, with score ranges becoming comparable to those in iteration 1. Five IndDoms showed near-zero variability, while two had ranges equal to or exceeding 0.1. Similarity boxplots, shown in A.9, continued to show moderate variability in most IndDoms in iterations 4 and 5, and a shift toward smaller score ranges in iterations 6 and 7.

8.3.3. Late Phase (Iterations 8 to 10)

The late phase is marked by clear improvements in realism and a final decline in similarity. Realism scores rose to a peak in iteration 9, where all 30 SSRSs received a score of 1.0, as confirmed by the corresponding boxplot. In the final iteration, the realism average declined slightly to 0.9 but exhibited the smallest score range among all iterations (excluding iteration 9), suggesting the highest output consistency. In comparison, similarity scores, after reaching a local high in iteration 7, began to decline and thus improve. The final iteration had the lowest average similarity score across the entire process, at 0.66. In addition to displaying the lowest and thus best score, the variability in iteration 10 was also reduced compared to earlier iterations, with a range from 0.5 to 0.8. The similarity heatmap in figure A.5 showed a substantial average score improvement in iteration 8, with 8 out of 10 IndDoms having scores of 0.8 or lower. However variability remained high, with scores ranging from 0.57 for E-commerce to 0.88 for Manufacturing. For this iteration, explicit prompt instructions were added to ensure greater SSRS diversity. In iteration 9, scores worsened slightly on average, with Logistics scoring highest at 0.9 and Education reaching a new domain-specific low of 0.57. This iteration also had a high score range of 0.33. The final iteration produced the best average similarity scores across all iterations combined also with low variability. The highest similarity score was 0.73 (Healthcare), and the lowest was 0.59 (Government), yielding the smallest range of 0.14. For this final iteration, insights from previous rounds were synthesized to construct a refined generation prompt version.

The subplots in figure A.7 show that similarity scores in iteration 7 fluctuated moderately, but with a smaller score range than in earlier iterations. In iterations 8 and 9, variability increased again. In iteration 10, similarity results were the most consistent on average, with lower score ranges and fewer fluctuations. Realism remained consistently high, and in iteration 9, all scores were perfect (1.0). The final iteration also showed minimal realism variation and high consistency. Realism boxplots shown in figure A.8 further confirm that iteration 9 achieved full score consistency across all IndDoms. Iteration 8 showed no variation in five IndDoms, compared to three in iteration 7. Iteration 10 followed closely behind iteration 9 in consistency, with minimal score variability across all IndDoms. Similarity boxplots shown in figure A.9 indicate a reversal of the trend seen in iteration 7: iteration 8 showed increased variability, while iteration 9 reverted to the pattern observed in iteration 1, with most IndDoms displaying low variability and only two to three IndDoms showing wider ranges. Iteration 10 exhibited slightly higher similarity variation within individual IndDoms but displays a much smaller score range across all IndDoms compared to earlier iterations.

8.4. Industry Domain-Level Insights

To better understand if and how results differ between different IndDoms throughout the ten iterations, an in-depth analysis of domain-specific trends was conducted in both degree of realism and semantic similarity scores. Figure A.4 displays the results for

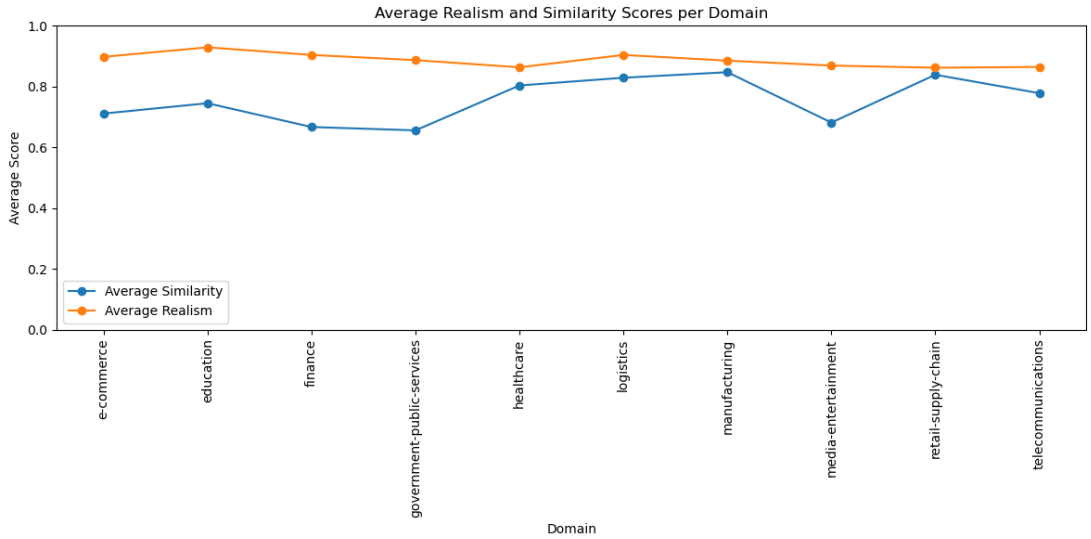

Figure 8.8.: The semantic similarity and degree of realism scores averaged over all ten iterations for each individual industry domain.

both metrics across all iterations for each individual industry domain while figure 8.8 displays the average scores across all iterations for each individual industry domain. For the variability of metric scores figures A.10 and A.11 give a detailed overview for each individual industry domain.

8.4.1. Realism Variability

Analysis of the realism score variability, displayed in figure A.10, across IndDoms and iterations reveals distinct consistency patterns. IndDoms such as Education, Finance, Healthcare, Logistics, Retail and Supply-Chain, and Telecommunications exhibit comparatively low variability throughout most iterations. In Education, the initial iterations demonstrate perfect or near-perfect consistency, with only a slight increase in range observed mid-cycle, particularly in iteration 7, followed by a return to near-zero variation in the later iterations. Finance shows near-perfect consistency in the first iteration and again from iterations 6 through 9. Although iteration 4 exhibits the highest variation for the Finance IndDom, the score range still remains under 0.05 which is comparatively low. For Healthcare, iterations 3 through 7 show comparatively modest variation with a score range of approximately 0.05. The final three iterations demonstrate near-perfect consistency, though iteration 10 shows a marginally larger range than its immediate predecessors. In Logistics, variation remains relatively stable during iterations 3, 4, 5, and 7. Iterations 6, 8, and 9 display near-perfect consistency, while iteration 10 introduces a marginal increase in variation compared to the prior two iterations, though still within a narrow range. Retail and Supply-Chain presents similar behaviour, with relatively equal variation in iterations 3, 5, 7, and 8, maintaining ranges around 0.05. Iterations 6,

9, and 10 exhibit near-perfect consistency. In Telecommunications, score variability is near-perfect in iterations 3 and 5 through 7 and again in iteration 9. Iteration 8 shows the highest variability in this IndDom, with a range exceeding 0.1. Iterations 4 and 10, in contrast, show minor variation with ranges around 0.05.

IndDoms demonstrating moderate to high variability in realism include E-commerce, Government and Public Services, Manufacturing, and Media and Entertainment. In E-commerce, early iterations present moderate variation, ranging from 0.05 to 0.15. Iteration 6 shows a temporary phase of zero variation, followed by a spike in variability during iteration 8. The final iterations, however, converge to very tight clustering with minimal range. Government and Public Services shows modest variation in iterations 3 through 5 and again in iteration 7, with ranges around 0.05. Iteration 6 has the highest variation in this IndDom, reaching a range of approximately 0.1. Variability significantly diminishes in iterations 8 and 9 and remains low, though slightly elevated, in iteration 10. Manufacturing exhibits low variation in iterations 3, 6, 9, and 10, with slightly higher ranges in the latter two. Other iterations in this IndDom show moderate variability, with iteration 7 reaching a maximum range slightly above 0.1. Finally, in Media and Entertainment, variability starts high at approximately 0.1 in the initial iteration. Iterations 7 and 9 return to near-perfect consistency, while iteration 10 sees a slight increase in range. Iteration 6 again exhibits elevated variability, comparable to the first iteration.

8.4.2. Similarity Variability

Similarity score variability, displayed in figure A.11 across IndDoms and iterations shows somewhat different trends. IndDoms such as E-commerce, Government and Public Services, Healthcare, and Logistics generally exhibit consistently low variability. In E-commerce, modest variation is observed in early iterations, followed by a minor increase mid-cycle. The final iterations show tight score clustering with minimal variation. For Government and Public Services, higher variation is evident in iterations 1, 2, and 4, each with a range of approximately 0.2. Iterations 3 and 9 display the most consistent scores. Iterations 5, 6, 7, and 10 maintain moderate variation, each with a range of roughly 0.1. In Healthcare, the highest variability appears in iteration 3 with a range of approximately 0.2. Iteration 5 exhibits the best consistency, while iterations 6 through 10 maintain modest variation with ranges near 0.1. The Logistics IndDom shows a gradual increase in variation from iteration 1 to iteration 10, culminating in the highest range at the end. Exceptions occur in iterations 7 and 9, both of which show notably less variation compared to other iterations.

IndDoms such as Education, Finance, Manufacturing, Media and Entertainment, Retail and Supply-Chain, and Telecommunications exhibit high or fluctuating variability in similarity scores. In Education, the first two iterations show high consistency. However, iterations 4, 6, and 8 exhibit substantial variation, while iterations 3, 5, and 9 show moderately elevated variability. Iterations 7 and 10 return to lower variability, though not reaching the level observed at the beginning. Finance begins with high consistency but demonstrates a peak in variation during iteration 5. Iterations 2, 3, 4, and 6 main-

tain modest variation around 0.2. Iterations 7 through 9 show reduced variation, but the final iteration again displays an increased range of approximately 0.2. In Manufacturing, iterations 1 and 2 exhibit little variation, and iterations 4 and 6 to 9 maintain low variability with slightly elevated ranges. However, iteration 10 exhibits the highest variation in this IndDom, with a range approaching 0.3. In Media and Entertainment, the first iteration shows good consistency, but iterations 2, 5, 7, and 8 exhibit significantly higher variation with ranges around 0.3. Variability decreases again in iterations 9 and 10. Retail and Supply-Chain starts with low variability in iteration 1, increasing until iteration 4, which shows the highest score range. Iterations 6 and 7 demonstrate the best consistency, with ranges comparable to the initial iteration. The final iteration again shows increased variability with a range of approximately 0.2. Telecommunications shows relatively low variation in iterations 1, 2, 4, 6, 8, and 9, each with ranges around 0.1. The highest variation occurs in iteration 3 with a range of about 0.2. Iteration 7 presents the most consistent results, while the final iteration shows the second highest range, slightly below 0.2.

8.4.3. Iterative Trend Patterns in Metric Scores

Trend analysis of realism and similarity scores across iterations reveals several recurring patterns. In certain IndDoms, similarity scores intersect or cross realism scores during the mid-iterations. This pattern is observed in Healthcare, Manufacturing, Retail and Supply-Chain, Telecommunications, and E-commerce. In Healthcare, realism begins at 0.88 in iteration 3 and remains stable in iterations 4 and 5, followed by a dip to 0.72 in iteration 6, then increasing to 1.0 in iteration 9, and ending at 0.91 in the final iteration. Similarity starts at 0.82, declines to 0.7 in the following iteration, rises to 0.93 by iteration 5, and then steadily drops to 0.73 by iteration 10. In Manufacturing, realism starts at 1.0 in iteration 3, drops to approximately 0.85 in iterations 4 to 7, reaches a low of 0.73 in iteration 8, and concludes at 0.91. Similarity begins at 0.96, falls to 0.83 in iteration 3, and stabilizes between 0.85 and 0.9 in subsequent iterations, aside from a low of 0.75 in iteration 5 and a final score of 0.67. In Retail and Supply-Chain, realism starts at 0.93–0.94, then decreases to 0.73 by iteration 7, rises to 0.78 in iteration 8, and finishes at 0.9. Similarity starts high at 0.95, drops to 0.76 in iteration 3, increases to 0.93 in iteration 7, and then declines to 0.7 by the final iteration. For Telecommunications, realism decreases steadily from 1.0 in iteration 3 to 0.75 in iteration 9, finishing at 0.9. Similarity begins at 0.88, drops sharply to 0.63 in iteration 4, increases to 0.95 in iteration 7, and then falls again to 0.71 in the final iteration. E-commerce shows relatively stable realism scores around 0.85 in iterations 3 to 5, dipping slightly to 0.8 in iteration 6, peaking near 1.0 in iterations 7 to 9, and ending close to 0.9. Similarity begins at 0.6, rises to 0.76 by iteration 4, falls to 0.6 in iteration 5, peaks at 0.87 in iteration 6, dips to 0.57 in iteration 8, rises again to 0.76, and ends at 0.67. Some IndDoms exhibit a persistent gap between similarity and realism. In Finance, realism begins at 1.0 in iteration 3, fluctuates between 0.8 and 1.0 until iteration 9, and ends at 0.89. Similarity starts at 0.95, declines to 0.68 in iteration 4, remains around 0.6 through iteration 6, briefly rises to 0.71 in iteration 7, and then drops again to 0.6 in the

final iteration. Education shows realism starting at 1.0, declining to 0.86 in iterations 4 to 6, rising to 1.0 in iterations 8 and 9, and dropping to 0.9 in iteration 10. Similarity begins at 0.92, then stabilizes around 0.75, drops to 0.6 in iteration 6, peaks at 0.82 in iteration 7, and finishes at 0.62. Government and Public Services mirrors the trend in Finance, with realism showing high and stable values and similarity dropping from 0.84 to around 0.6, with minor peaks at 0.7 in iterations 6 and 8, and a final value of 0.59. In Media and Entertainment, realism is steady at 0.86 initially, drops to 0.65 in iteration 6, and returns to approximately 0.9 afterward. Similarity starts at 0.88, drops sharply to 0.6, rises to 0.74 by iteration 5, then fluctuates between 0.6 and 0.7, and ends at 0.62. The Logistics IndDom is a notable case where realism and similarity scores are closely aligned. Realism begins at 0.93, remains around 0.9 through iteration 6, peaks at 0.97 in iteration 7, dips to 0.8 in iteration 8, and ends at 0.89. Similarity starts at 0.85, remains between 0.8 and 0.9 for most of the process, with lows of 0.77 in iteration 5 and 0.71 in the final iteration.

9. Evaluation by Human Experts

This chapter presents the results of a human IndDom expert study conducted to assess the realism of SSRSs generated by the SSRS-Gen process. While previous chapters focused on automated evaluation using LLM-based metrics, this study provides an external validation of the realism criterion using domain-specific human judgment. A total of 60 participants took part in the evaluation, with responses collected across all ten industry domains included in the SSRS-Gen process. Each expert was asked to assess at least one complete SSRS. For every instantiated section of the SSRS template, participants assigned a binary label of "realistic" or "unrealistic". Following this section-wise evaluation, participants provided an overall realism judgment on a 5-point Likert scale. These two levels of granularity enable both detailed and holistic validation of the generated outputs. The following sections detail the structure of the questionnaire, report key findings from the section-level and overall ratings, and analyse patterns across domains.

9.1. Questionnaire Study Design and Evaluation

This section details the design and implementation of the expert study conducted to validate the realism of SSRSs generated in the final iteration of the SSRS-Gen process. As previously established, realism represents the most critical quality dimension for evaluating SSRS usefulness in downstream tasks. To assess this, a small-scale questionnaire study was administered, focusing exclusively on expert judgments of realism without requiring participants to evaluate completeness or other structural features. The following describes the study's design, participant recruitment strategy, questionnaire structure, and pre-test adjustments.

The goal of the expert study was to conduct an initial realism evaluation of the SSRSs generated in the last iteration. Realism is the most critical metric, as unrealistic SSRS are unsuitable for any potential downstream tasks. The study was administered via an online questionnaire using SoSci Survey and was conducted in a fully anonymous manner. Only non-identifying metadata was collected, focusing on participants' professional background and self-assessed domain expertise. To be eligible for participation, individuals were required to have prior professional experience with software projects. Additionally, participants were asked to indicate whether they considered themselves technical experts, domain experts, or both, how many years of work experience they have, and to specify the industry domains in which they felt most confident to evaluate an SSRS. The list of ten IndDoms used in the process execution was provided, and participants were required to select at least one they feel confident in. Those who selected "None of the listed domains" or reported no relevant experience were excluded from

the study. Each participant was then randomly assigned to one of the their selected IndDoms. For each of the ten IndDoms one of the three generated SSRS was chosen at random for evaluation in the study. In total this results in a fixed sample of 10 out of 30 SSRS. This sample size was deliberately chosen in order to keep the study feasible in terms of the overall time frame and the manual effort required to analyse the retrieved data set within the scope of this thesis. The SSRS evaluation questionnaire was structured around the four sections of the SSRS template: System Overview, Functional Requirements, Non-Functional Requirements, and Constraints. For each element of the four sections, participants were asked to indicate whether the content appeared to them as rather "Realistic" or rather "Unrealistic" using a binary radio button. The default selection was set to "Unrealistic". If a section was marked as "Unrealistic," an optional text field was provided for participants to write a justification. No response fields were mandatory. The four SSRS sections and their individual elements were presented in the same order as specified in the SSRS template and were not randomized. After completing all four main sections section, the participant was asked to provide a realism assessment of the SSRS as a whole using a Likert-Scale from 1 to 5 labelled with "Very Artificial" to "Very Realistic" as well as a "Not Sure" option and an optional text field for justification.

Before starting the study an initial pre-test was conducted with five participants spread across four IndDoms. Based on the insights an additional page was added before the actual questionnaire which contained two information boxes that a participant has to confirm before being able to proceed. First that they have to label each section of a given SSRS with either "Realistic" or "Unrealistic" and that these options reflect tendencies (rather realistic vs. rather unrealistic), not absolute right or wrong answers. Second that they should evaluate only the realism of the information provided, not its completeness, because the content follows a predefined template that limits the scope of included information. After confirmation the main questionnaire then starts. These additions were made because it was observed that participants in the pre-test frequently remarked missing requirements or information that they would expect in an SSRS but which are not defined in the SSRS template.

Initially participants were also asked to specify their specific job role, but this question was replaced after the pre-test by the domain and/or technical expert question, because for this small-scale study there is little benefit or use for retrieving this data. The study was estimated to reach around 50 participants and thus would not have a large enough data set to gain statistically significant insights on differences in realism perception based on job role. In the pre-test the average completion time of the whole study was approximately 15 minutes which was then subsequently communicated to the participants of the main study as the expected completion time. Participants were then recruited through purposive sampling and snowball sampling with a combination of professional and personal networks, mailing lists such as the SWT GI specialists group and targeted outreach via LinkedIn. This recruitment strategy was intended to reach practitioners with relevant technical and/or domain-specific experience. Prior to beginning the questionnaire, participants were required to provide explicit informed consent.

Industry Domain	No. of Evaluations
Finance	21
E-Commerce	10
Education	8
Government	5
Logistics	4
Manufacturing	4
Healthcare	3
Media-Entertainment	2
Telecommunications	2
Retail-Supply-Chain	1

Table 9.1.: Number of evaluations for each industry domain.

This small-scale study is intended as a preliminary validation of the results from the last iteration. While it offers early insights into the perceived realism and practical plausibility of the generated SSRS, it does not constitute a comprehensive evaluation. A more extensive follow-up study is proposed as a direction for future research.

9.2. Expert Evaluation Results

This section presents the results of the expert study conducted to assess the perceived realism of the SSRSs generated in the final iteration of the SSRS-Gen process. The evaluation focused exclusively on realism, as it represents the most critical dimension for determining the practical utility of SSRSs in downstream applications. The results are structured into four subsections. First, an overview of the participant distribution is provided, including the number of responses per industry domain. This is followed by a detailed analysis of the binary section-level realism ratings, which reflect how individual SSRS elements were judged across the four main SSRS template sections. The third subsection reports on participants' overall realism assessments using a five-point Likert scale. Finally, selected qualitative justifications are analysed to offer descriptive insights into recurring themes or concerns raised by participants. Together, these results provide a multifaceted view of how domain-experienced practitioners perceived the realism of the generated SSRSs.

9.2.1. Participation Overview

The study was conducted over a period of one month. In total 60 questionnaires were answered by 52 participants, as two participants did the maximum of three evaluations and four participants evaluated a second SSRS. The number of SSRS evaluations differs greatly across Industry domains. As shown in table 9.1 the IndDom of Finance has by far the most evaluations with 21 followed by E-Commerce with 10 and Education with

8. All other IndDoms have 5 or less evaluations with Retail-Supply-Chain only having a single one.

9.2.2. Participant Expertise

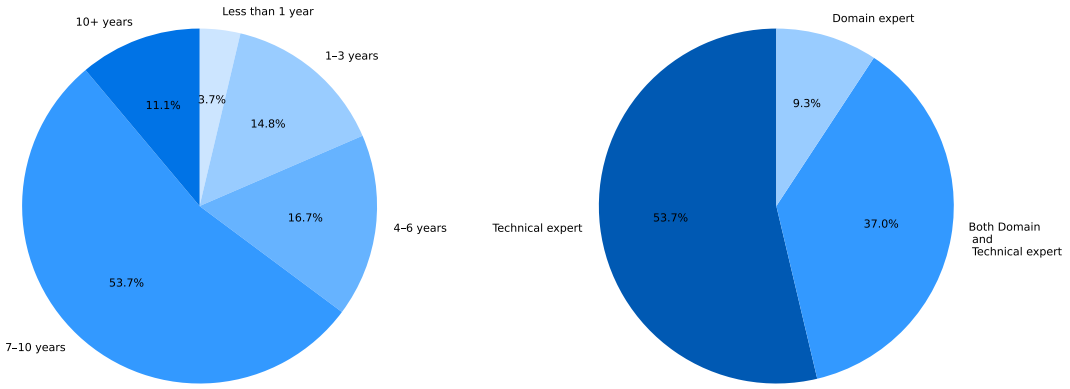

Figure 9.1.: Participants self-reported expertise, including years of professional experience and type of expertise.

To contextualise the expert evaluations, this subsection provides an overview of the professional background and self-assessed expertise of the study participants. Understanding participants' experience and expertise levels is essential for interpreting the realism assessments and identifying potential sources of bias or variation in judgement.

Figure 9.1 displays two pie charts, one for the years of experience and one for the expert type, as self-reported by each participant. The first pie chart on the left presents the years of professional experience. Most participants reported substantial experience, with 53.7% having between 7 and 10 years, and 11.1% reporting more than 10 years. Participants with 4–6 years of experience made up 16.7%, while 14.8% had 1–3 years. Only a small fraction (3.7%) had less than one year of experience. This data suggests that the sample is composed primarily of experienced professionals, supporting the credibility of the realism judgments provided.

The second pie chart on the right illustrates the distribution of participants by type of expertise. A majority of respondents identified as technical experts (53.7%), while 37% indicated expertise in both technical and domain-specific areas. The remaining 9.3% considered themselves domain experts only. This mix suggests a participant pool with predominantly technical or hybrid expertise.

9.2.3. Section-Level Binary Evaluation Results

Each SSRS questionnaire was composed of 17 predefined sections as specified in the SSRS template introduced in section 3.4. These sections were independently evaluated by participants for their perceived realism using a binary assessment as described in

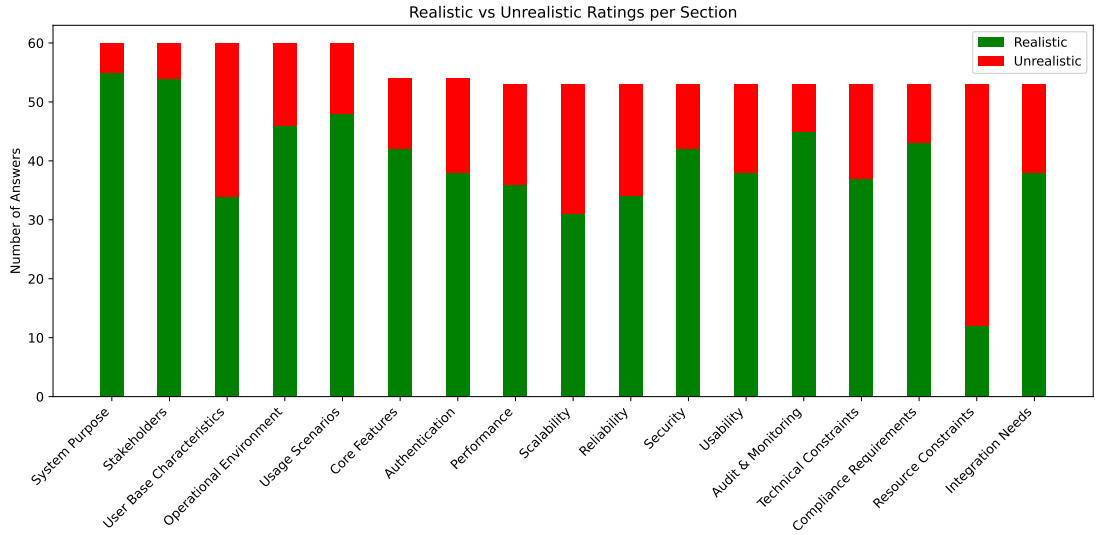

Figure 9.2.: Realism ratings per template element, showing the absolute number of responses marked as realistic or unrealistic.

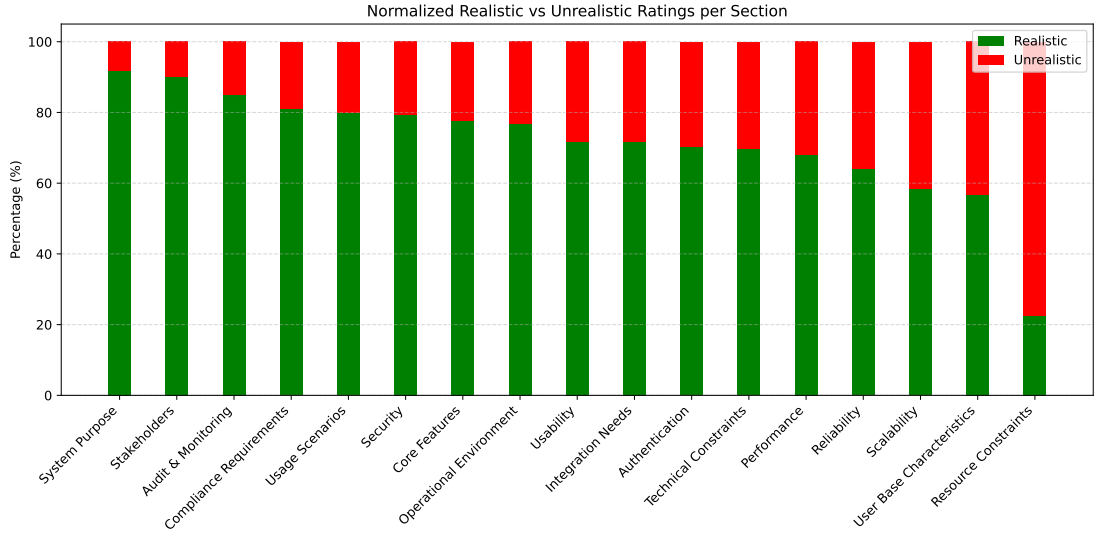

Figure 9.3.: Realism ratings per template element, normalized and sorted by proportion of responses marked as realistic.

the previous section. The sections System Purpose and Domain/Context were assessed jointly as one evaluation item, as were both authentication-related requirements under Functional Requirements. All other sections were rated individually. The elements of the System Overview category received a full 60 responses. The two Functional Requirements items were evaluated 54 times, while the remaining sections under Non-Functional Requirements and Constraints categories were rated 53 times each. Thus in total there are 53 complete datasets, and seven partially complete datasets. All available datasets were included in the analysis, as each SSRS template element was evaluated independently. This independence allows for the use of partial responses without introducing any dependency-related bias.

The resulting dataset is displayed in figures 9.2 and 9.3. Figure 9.2 displays the absolute count of assessments for each individual SSRS element while figure 9.3 normalizes the data to 100% of retrieved answers for the given element. For ease of comparison figure 9.3 is also sorted by the proportion of "Realistic" ratings. Both figures clearly show that the individual SSRS elements exhibit varying proportions of ratings marked as "Realistic". It directly stands out that the elements "System Purpose" and "Stakeholders" received the highest realism assessment with over 90% of answers marking it as "Realistic. In comparison the element "Resource Constraints" by far received the least "Realistic" ratings with only 23%. Across all elements the average proportion of "Realistic" ratings was 0.71. When grouped by their respective SSRS template categories, System Overview received the highest average realism score at 0.79, followed by Functional Requirements with 0.74, Non-Functional Requirements at 0.71, and Constraints at 0.61. The individual sections with the highest realism ratios were System Purpose (0.92), Stakeholders (0.90), Audit & Monitoring (0.85), and Compliance (0.81). In contrast, the lowest realism ratings were observed for Resource Constraints (0.23), User Base Characteristics (0.57), Scalability (0.59), and Reliability (0.64). The standard deviation from the overall mean realism percentage across all 17 sections was 0.15, which suggests that most sections had realism proportions that fell within the interval of approximately 0.56 to 0.86, reflecting moderate dispersion around the mean and indicating that while realism was generally perceived positively, some sections showed significantly higher or lower realism consistency.

9.2.4. Overall SSRS Realism Assessment Results

To assess the overall perceived realism of each SSRS, participants were asked to provide a rating using a 5-point Likert-scale with the options: *Very Artificial, Somewhat Artificial, Neutral, Somewhat Realistic*, and *Very Realistic*. Out of 60 total responses, nine participants did not answer this question, and none selected "Not Sure" so in total there are 51 overall assessments. The absolute distribution of responses is depicted in figure 9.4, which shows eight participants rated their given SSRS as "Very Realistic", 26 as "Somewhat Realistic", nine as "Neutral", four as "Somewhat Artificial", and two as "Very Artificial". In figure 9.4 there are only two participants displayed that did not answer this question instead of actual number of nine missing answers. This is, because those participants exited the study before navigating to the last page of the

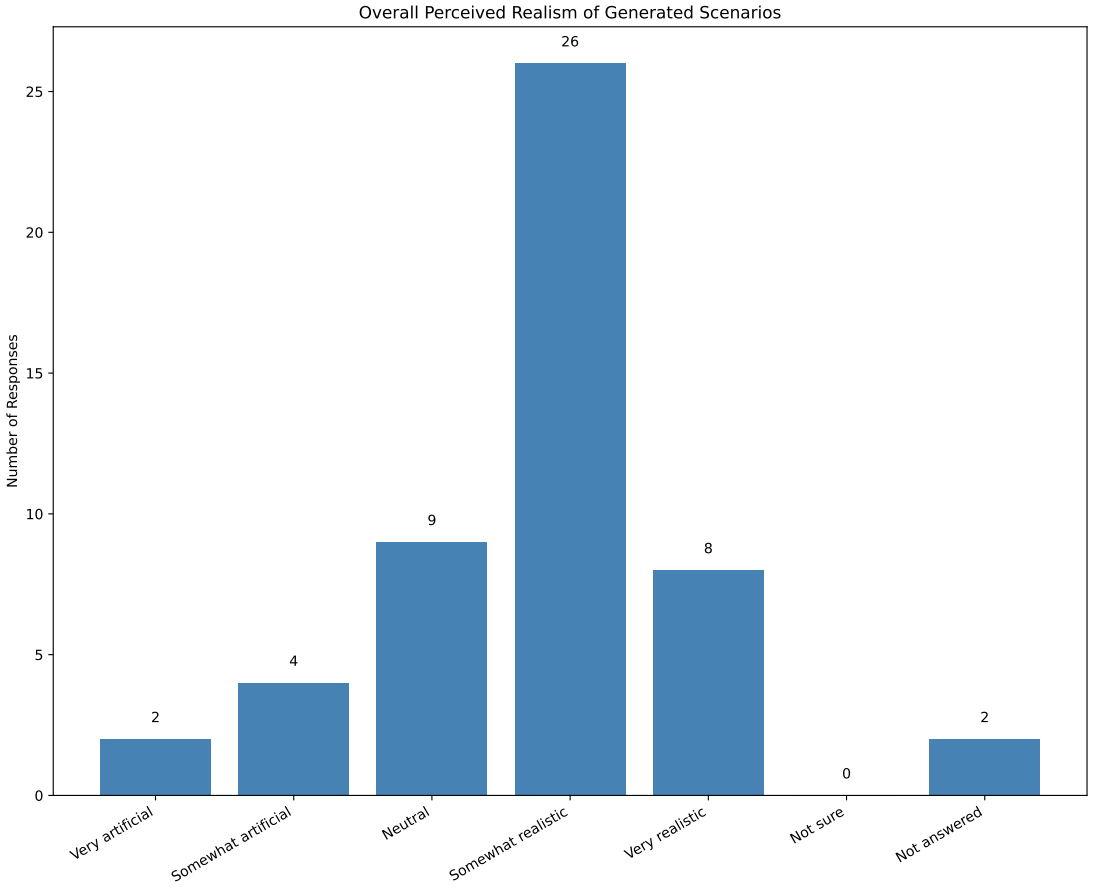

Figure 9.4.: Distribution of participant responses for the overall realism rating of the SSRS.

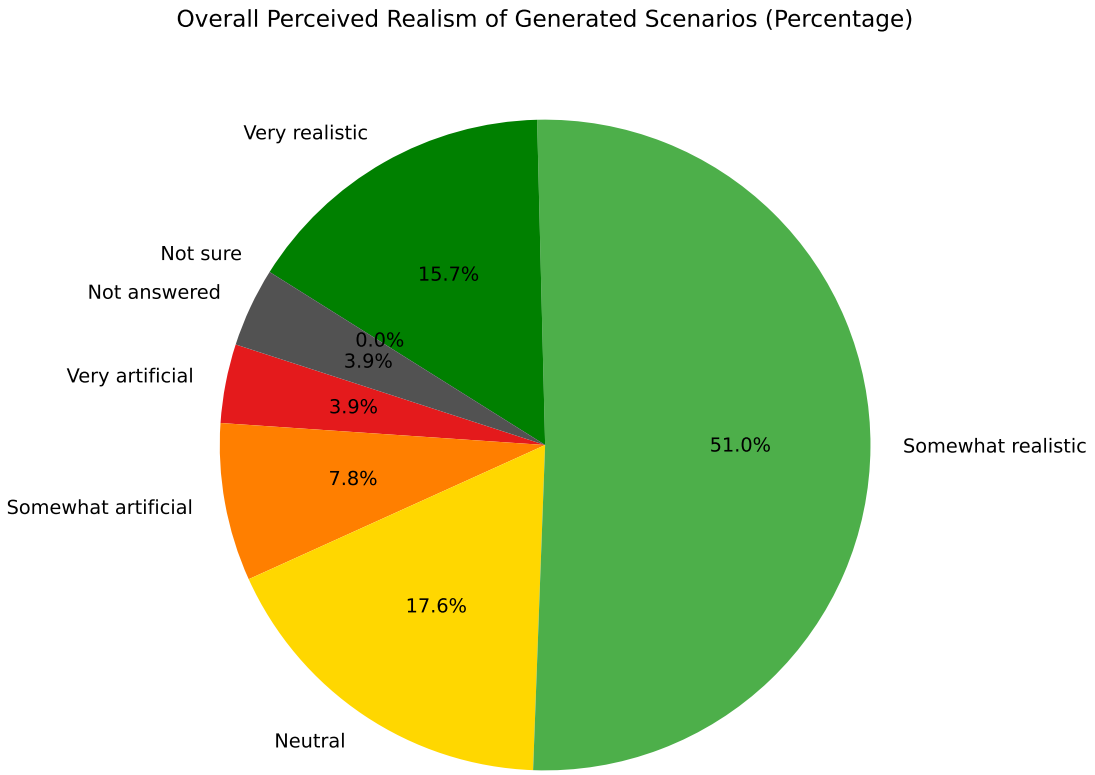

Figure 9.5.: Proportion of participant responses for the overall realism rating of the SSRS.

questionnaire. This distribution shows that a majority of participants (34 out of 51; ∼67%) viewed the SSRSs positively, selecting one of the two highest realism levels. In contrast, only six out of 51 responses (∼12%) reflected a negative perception, and ∼18% of respondents assessed the SSRS as neither particularly artificial nor realistic. The corresponding bar chart illustrates that moderately positive realism ratings were most common, while extreme responses on either end of the scale were relatively rare.

The pie chart in figure 9.5 further reinforces this trend by visually emphasizing the overall positive perception of the generated SSRSs. These findings suggest that the generation methodology was generally effective in producing SSRS that were perceived as realistic by participants. Nonetheless, the presence of neutral and critical responses indicates that further refinements may be required to consistently align with expert expectations.

9.2.5. Descriptive Overview of Justification Responses

To complement the quantitative realism ratings, participants could provide optional textual justifications for any SSRS section they evaluated as "Unrealistic." While comments varied in depth and specificity, a thematic analysis revealed several recurring concerns. This qualitative data provides important insights into realism issues of generated SSRS and allow for a direct comparison with realism issues identified by the LLM itself through its Self-Assessment. The following paragraphs highlight the most important remarks made by the IndDom Experts.

Oversimplification and Generic Phrasing

A substantial number of comments indicated that the requirements appeared overly generic, undermining their perceived authenticity. These responses suggest that generated SSRSs sometimes failed to capture the complexity and variability typical of real-world specifications. For example, one participant remarked: "Look like textbook examples", while another commented: "It's very generic or broad".

Lack of Detail and Ambiguity

Another common theme was insufficient specificity. Participants flagged vague statements that lacked the technical depth needed to assess plausibility. One respondent wrote: "I would expect some more detail here", and another noted: "That is very vague. [...] more information [...] should be defined". These responses point to a gap between expected and actual granularity in domain-specific content.

Logical Incoherence Between Requirements

Another issue raised by participants was the presence of internal contradictions within some SSRSs. These comments highlighted inconsistencies between different requirements or contextual assumptions. For instance, one expert questioned the applicability of European data protection laws in a non-European context, stating: "gdpr is for europe, which wasn't a target (unlike americas and africa)". Another similarly remarked: "I am not sure if the GDPR regulations are realistic here, as the tool "Focuses on North America, South America, and regions in Africa where hybrid education is rapidly growing." Would GDPR be required there?" Such feedback reveals the presence of logical incoherence between individual requirements.

Overly Optimistic and Ambitious Requirements

Several expert comments pointed to requirements that appeared overly optimistic or ambitious. These responses suggest that some SSRSs include assumptions that may not hold under realistic conditions, reflecting an underestimation of complexity or overconfidence in execution capacity. One participant for example noted: "The 12 month with a total staff of 30 is very optimistic," highlighting concerns about the feasibility of delivery

within the proposed constraints. Another expert remarked: ”15.000 users for an initial round seems too ambitious,” pointing to unrealistic expectations about user adoption rates.

10. Related Work

Having presented both the data gathered during the execution of the SSRS-Gen process and the results of expert study, this chapter positions the thesis within the broader context of AI-supported requirements engineering, with a particular focus on the emerging role of LLMs in automating the generation and analysis of software requirements.

The application of AI and especially LLMs in software requirements engineering (RE) has gained increasing attention, particularly as researchers explore ways to automate traditionally manual, expert-driven activities. Prior studies have demonstrated the potential of LLMs to assist with tasks such as requirements elicitation, classification, transformation, and even generation. Additionally, several machine learning and natural language processing (NLP) pipelines have been proposed to extract or structure requirements from stakeholder input or user feedback. This chapter reviews and analyses a representative set of these works to position this thesis in the broader research landscape. Each reviewed study is evaluated in terms of its objectives, methods, use of LLMs, and treatment of software requirements, followed by a summary that highlights how this thesis extends and differentiates itself from the existing body of work.

10.1. Traditional AI Techniques in Requirements Engineering

Before the rise of large language models, a wide range of traditional AI techniques, particularly machine learning, deep learning, and classical NLP, were applied to automate various tasks in requirements engineering. These approaches focused largely on extracting, classifying, or summarizing requirements based on structured or semi-structured data sources such as user feedback or meeting transcripts. While these methods laid important groundwork for automation in RE, they often required domain-specific training data, relied on predefined rules or classifiers, and typically addressed only narrow subtasks such as feature suggestion or requirement categorization. The following two works illustrate representative approaches from this category and serve as a conceptual foundation for understanding how newer LLM-based methods differ.

10.1.1. Generating Requirements Out of Thin Air: Towards Automated Feature Identification for New Apps

The work by Iqbal et al. [ISM19] presents a conceptual framework for automatically generating software requirements for new mobile applications by mining app store content and competitor feedback. The approach is motivated by the difficulty of engaging users or domain experts in early-stage software projects. It proposes an automated pipeline

to identify relevant features through data mining, sentiment analysis, and clustering, aiming to infer what a new app should support by learning from existing ecosystems.

This study aligns closely with the motivational framing of the SSRS-Gen process. Both works seek to address scenarios where real-world requirement specifications or expert input are unavailable, and both aim to generate substitute artefacts that can support downstream software processes such as validation, testing, or recommendation. However, the techniques differ fundamentally. Iqbal et al. propose a pipeline based on traditional NLP and machine learning, whereas SSRS-Gen leverages zero-shot prompting with LLMs to generate fully structured SSRSs directly.

Importantly, the work by Iqbal et al. also explores the use of AI to generate requirements, using NLP and traditional machine learning techniques to identify discrete feature candidates from unstructured user feedback. In contrast, this thesis investigates whether and how LLMs can be applied to address similar goals. Compared to their focus on extracting isolated features, SSRS-Gen generates entire structured specifications, including system overviews, functional and non-functional requirements, and constraints. Furthermore, SSRS-Gen integrates automated quality evaluation and iterative prompt refinement, expanding the scope from feature-level suggestions to the generation and assessment of complete requirement specifications.

In summary, this work is highly relevant as conceptual precedent and motivational support for automated requirement generation in data-scarce settings, but differs in both granularity and technical approach from the LLM-driven strategy proposed in this thesis.

10.1.2. Requirements-Collector: Automating Requirements Specification from Elicitation Sessions and User Feedback

The Requirements-Collector tool, proposed by Panichella and Ruiz [PR20], automates the process of extracting and classifying requirements from elicitation session transcripts and user reviews. It combines machine learning (ML) and deep learning (DL) techniques to identify and categorize functional and non-functional requirements, and ultimately produce structured outputs like user stories. The tool aims to reduce the manual effort typically involved in analysing stakeholder input and user feedback.

This work is relevant to the thesis in that it also targets automation of requirements engineering tasks using AI, particularly in data-intensive and time-constrained contexts. However, the approach relies on traditional ML/DL models, not LLMs. Its focus is on extracting requirements from human-generated content, whereas the SSRS-Gen process creates entirely SSRSs using zero-shot LLM prompting without any human-generated source material.

Despite this methodological difference, Requirements-Collector provides a useful contrast. It highlights the limits of conventional ML pipelines, such as the need for labelled training data, lower flexibility across domains, and limited support for structured document generation, which the SSRS-Gen process addresses by leveraging LLMs' generalization and text generation abilities.

In summary, this work is valuable for contextualizing the evolution from ML-based automation of RE tasks toward LLM-based generative approaches like the one proposed in this thesis. While it does not employ LLMs itself, it represents a significant predecessor in the push toward automated requirement pipelines.

10.2. LLM-Driven Approaches to Requirements Engineering

With the emergence of LLMs, the landscape of AI-assisted requirements engineering has evolved significantly. These models provide general-purpose language understanding and generation capabilities, enabling zero-shot or few-shot application to RE tasks that previously required domain tuning or extensive training data. Recent research has explored how LLMs can support requirement generation, assessment, classification, and transformation. This section reviews key studies that apply LLMs to RE, highlighting how they expand the scope of automation and how they compare to the SSRS-Gen process proposed in this thesis.

10.2.1. An Automated Model of Software Requirement Engineering Using GPT-3.5

This work by Yeow et al. [YRA24] investigates the use of GPT-3.5 to automate parts of the requirements engineering process, specifically the generation of survey and interview questions for requirements elicitation across different industrial domains. The authors evaluate the linguistic quality and contextual relevance of GPT-generated questions using metrics like Flesch Reading Ease and expert judgment. Their findings confirm that GPT-3.5 can produce coherent, domain-aligned elicitation artefacts, although the results vary depending on the specificity of the prompts and domain.

This study shares a key motivation with the present thesis: addressing the lack of accessible, high-quality domain-specific requirements artefacts through LLM-based generation. However, Yeow et al.'s focus lies in supporting the requirements elicitation process for a known, predefined idea of a system such as a "banking system" as they show in their example prompt [YRA24]. Their prompts to GPT-3.5 are framed around generating survey and interview questions tailored to specific, named systems, with the intent of collecting stakeholder input. In contrast, this thesis is concerned with generating the requirements themselves, and not for an existing system, but for an entirely new and synthetic system that the LLM is asked to invent within a given industry domain. In this sense, the prior work addresses the preparation for the actual requirements documentation, while SSRS-Gen targets the creation of the documentation itself, including all functional, non-functional, and contextual elements, without assuming any predefined software scope.

Another notable distinction lies in evaluation methodology. Yeow et al. rely largely on readability metrics and manual inspection, whereas this thesis introduces a multi-metric evaluation pipeline combining LLM-based self-assessment, addressing the defined process requirements of *R2-Comparability*, *R1-Realism*, and *R3-Diversity*, as well as a follow-up

expert study focusing on *R1-Realism*. In summary, while the two approaches differ in their specific outputs and use cases, they share a broader objective: demonstrating that LLMs can function as viable supports for expert-driven RE tasks in contexts where data is limited or unavailable.

10.2.2. Improving Requirements Completeness: Automated Assistance through Large Language Models

In this paper, Luitel et al. [LHS24] explore the use of BERT, a transformer-based language model, to improve the external completeness of natural language requirements. By masking parts of existing requirements and analysing BERT's predictions for plausible missing terms, the approach aims to identify omissions and enhance the clarity and thoroughness of requirement documents. The proposed system combines masked language modelling with domain-specific corpora and machine learning classifiers to filter and validate BERT's suggestions.

This study is particularly relevant to the evaluation phase of the SSRS-Gen process. While the generation strategies differ as BERT is used to assess and enhance existing human-written requirements, not to create them, both works use LLMs as external knowledge sources to approximate what might be missing in a requirement. A key distinction lies in the modelling approach: this paper uses BERT in a white-box fashion, whereas SSRS-Gen operates with ChatGPT in a black-box setting, using zero-shot prompting and self-assessment techniques to generate and evaluate full documents. Nonetheless, both works validate that pre-trained LLMs can act as substitute for expert judgment in RE quality assessment, especially when direct access to domain specialists is limited.

In summary, this paper complements the SSRS-Gen approach by demonstrating how LLMs can be used not only to generate, but also to systematically evaluate and enhance the completeness of software requirements, supporting the credibility of automated RE assistance in practice.

10.2.3. Assessment of ChatGPT's Proficiency in Software Development

The study by Kim et al. [Kim+23] evaluates ChatGPT's capability to support tasks across the entire software development lifecycle, including requirements analysis, modelling, and implementation. Within the requirements phase, the authors prompt ChatGPT to identify ambiguities, generate functional and non-functional requirements, and produce use case specifications based on an informal problem description. They also assess the coherence and completeness of these outputs and discuss the tool's strengths and limitations in mimicking expert behaviour.

This work is relevant to this thesis, as it validates the ability of LLMs to generate structured requirement artefacts, not just support ideation or brainstorming. However, their focus is more exploratory and project-based: a single predefined system is used as a case study, and the evaluation remains qualitative and task-centric. In contrast,

this thesis introduces SSRS-Gen as a process intended to be generalisable and domain-independent for systematically generating and assessing SSRS across multiple industry domains using an LLM, though this remains to be empirically verified.

Additionally, while Kim et al. rely on manual expert evaluation and informal judgments, the SSRS-Gen process integrates automated, structured self-assessment grounded in scientifically evaluated prompt engineering techniques. This allows for scalable evaluation and refinement without constant human expert involvement, which is a key differentiating aspect of this thesis in data-scarce settings.

In summary, while both works investigate the feasibility of LLM-based requirements generation, this thesis builds on and extends that idea by introducing a repeatable, self-assessing generation pipeline with custom evaluation metrics and iterative prompt refinement.

10.2.4. Extracting Domain Models from Textual Requirements in the Era of Large Language Models

This paper by Arulmohan et al. [AMM23] investigates the use of GPT-3.5 to extract domain-specific elements from unstructured user stories, with the goal of transforming informal requirements into structured domain models. The authors compare the GPT-based extraction pipeline to a rule-based tool and a supervised NLP model, using an annotated benchmark dataset for evaluation.

This study is relevant for this thesis, because it applies LLMs to transform unstructured RE artefacts into structured outputs, a goal that aligns with the template-based SSRS generation process proposed here. Both works leverage prompt engineering and constrained outputs to guide LLM behaviour, reinforcing the value of prompt structure in controlling LLM output quality.

However, their goal is transformational, focusing on restructuring existing user generated stories, whereas SSRS-Gen is generative, producing full requirement specifications from scratch. Additionally, while this paper compares LLMs to white-box NLP tools using benchmark datasets, SSRS-Gen emphasizes self-assessment and prompt refinement in zero-shot settings where no annotated data is available.

In summary, this work strengthens the thesis's positioning by showing that LLMs outperform traditional tools in structuring requirements-related content, even though the two works target different phases of the requirements engineering pipeline.

10.2.5. NLP4ReF: Requirements Classification and Forecasting – From Model-Based Design to Large Language Models

The NLP4ReF framework by Peer et al. [PMR24] presents two distinct approaches for automating requirement classification and forecasting: one based on traditional NLP and machine learning (NLP4ReF-NLTK), and another leveraging large language models (NLP4ReF-GPT) to generate new, potentially overlooked requirements from an existing set.

It compares a traditional NLP pipeline with a GPT-3.5-based alternative. The GPT model is used both to classify requirements into Functional Requirements (FR)/ Non-Functional Requirements (NFR) categories and to generate additional requirements that complement the input set. The work includes both quantitative metrics and human evaluations of the generated outputs.

This paper is relevant to the present thesis for several reasons. First, it is one of the few studies to explicitly explore requirement generation using ChatGPT. Second, their use of automated quality metrics and expert validation is comparable to SSRS-Gen. However, unlike this thesis, which applies structured prompt design techniques to guide LLM behaviour, their approach does not explicitly incorporate prompt engineering methods.

Additionally, the framing of NLP4ReF differs in scope. It assumes the presence of a seed list of requirements, from which GPT generates additional ones. By contrast, the SSRS-Gen process generates requirements entirely from scratch, based only on an industry domain and a structured prompt. Moreover, SSRS-Gen extends beyond requirement statements to produce full SSRSs, covering system overview, constraints, and both FR and NFR dimensions in a format adapted from ISO 29148 [ISO18].

In summary, NLP4ReF offers a strong technical and conceptual precedent for this thesis, particularly in its use of GPT for requirement generation and evaluation. It complements SSRS-Gen by demonstrating that LLMs can enrich existing RE artefacts, while this thesis shows they can also fully generate and iteratively refine those artefacts in the absence of any ground truth.

10.3. Summary and Distinction from Related Work

While the reviewed literature demonstrates growing interest in applying AI techniques to various aspects of requirements engineering, this thesis makes a distinct contribution by introducing a structured, end-to-end process for the zero-shot generation and self-assessment of full Synthetic System Requirement Specifications. Compared to both traditional machine learning pipelines and more recent LLM-based approaches, the SSRS-Gen process offers a unique combination of generative scope and built-in evaluation.

Traditional AI-based approaches, such as those by Iqbal et al. [ISM19] and Panichella and Ruiz [PR20], focus on automating classification and feature extraction tasks from elicitation sessions or user feedback using conventional NLP or supervised learning techniques. While they also pursue the automation of requirements-related tasks, their methods are limited to isolated aspects, such as identifying discrete feature candidates, and depend heavily on training data and structured input. In contrast, SSRS-Gen generates complete and cohesive requirement documents from scratch and operates in zero-resource settings, leveraging pre-trained LLMs without domain-specific fine-tuning.

More recent work explores the use of LLMs to support requirement generation, transformation, and evaluation. For example, Yeow et al.[YRA24] and Kim et al.[Kim+23] use LLMs to generate survey questions, use case specifications, or classify RE tasks, but their approaches focus on predefined systems or isolated artefacts and lack a repeatable

evaluative pipeline. In contrast, SSRS-Gen is designed to generate entire SSRS documents for entirely new systems and includes automated self-assessment mechanisms for completeness and realism.

Works like NLP4ReF (Peer et al. [PMR24]) come closer in spirit by using GPT to forecast new requirements. However, they depend on existing partial input sets, whereas SSRS-Gen does not assume any prior system specification. Furthermore, while Peer et al. evaluate their outputs using automated metrics and expert validation, SSRS-Gen additionally employs structured prompt engineering and LLM-based self-assessment, extending the scope of automation to include both document generation and quality control.

Finally, while many reviewed papers rely on human judgment or basic readability metrics for evaluation, SSRS-Gen introduces scalable, automated assessment using self-assessment prompting. This enables evaluation for both comparability and realism without requiring continuous expert oversight.

In summary, this thesis distinguishes itself by proposing a prompt-engineered generation process for SSRSs that functions in zero-resource contexts, integrates LLM-based quality self-evaluation, and shifts the focus from narrow RE tasks to the automated production of structured, full-scope requirement specifications.

11. Discussion

This chapter synthesises the key findings of the SSRS-Gen process, drawing on both the iterative prompt refinement results and the external expert evaluation. It begins by analysing the observed effects of individual prompting strategies and design choices, followed by a detailed assessment of how the model's self-evaluation capabilities align with human expert judgments. The subsequent sections provide a broader review of the SSRS-Gen process, reflect on the terminology used throughout, and highlight critical limitations of the approach. Finally, directions for future research are outlined to guide continued development and validation of the proposed SSRS-Gen process.

11.1. Observed Impact of Prompting Strategies

The SSRS-Gen process underwent ten iterations of prompt refinement, during which a variety of prompting strategies were tested, adjusted, and evaluated for their effect on output quality as measured by the defined metrics. This section discusses the insights gained from those iterations, focusing on the observed impact of specific prompt patterns on the quality of generated SSRS. In addition to pattern-specific evaluations, some general insights into prompt design are presented to inform future prompt engineering practices in complex generative tasks.

11.1.1. Template Pattern

From the outset, the use of the template pattern in the form of the predefined SSRS template was highly effective in ensuring that generated SSRSs adhered to the expected structure. Even without complex instructions or patterns as in the first iterations, the usage of the pattern led to complete format adherence across all outputs. Across all ten iterations there was not a single SSRS that was incomplete. This result indicates that the template pattern is an effective and consistent measure in zero-shot settings to accurately communicate the desired output format and content to an LLM.

11.1.2. Persona Pattern

The introduction of an expert persona in Iteration 3 coincided with a further drop in similarity scores and a notable decrease in similarity score variability, suggesting more consistent diversity across outputs. However, its specific impact on the DoR score remains difficult to isolate, as in this iteration the change from boolean to decimal realism scoring was conducted and thus there is no comparison to previous scores. Thus

the exact effect of the persona pattern for the realism of generated SSRS cannot be clearly determined from the available data.

In contrast, when the persona pattern was introduced in the realism self-assessment prompt in iteration 4, its effects were more directly observable. The addition led to a noticeable drop in DoR scores but also resulted in more detailed and critical evaluations of SSRSs. Moreover, DoR score variability decreased, indicating more consistent assessments across outputs. These effects suggest that the persona pattern improved the depth and reliability of realism self-assessments by framing the model's judgment through a more domain-aware and critical lens. This points to a potential design insight: combining an expert persona with a self-assessment prompt may be an effective strategy for eliciting more nuanced and consistent quality evaluations from an LLM.

11.1.3. Self-Criticism

Three distinct forms of self-criticism were employed in the SSRS-Gen process: self-assessment for completeness, self-assessment for realism, and self-refinement during SSRS generation. The completeness self-assessment, which involved verifying whether all points of the template were fully instantiated, proved highly effective. As a straightforward structural check, it leveraged the LLMs strength in pattern matching and consistently produced reliable results.

The realism self-assessment was initially implemented in a boolean form during Iterations 1 and 2, but all outputs were marked as "realistic," offering only limited insights. Later iterations adopted a more granular scoring approach using a decimal score, which revealed useful observations. However self-assessments often focused extensively on resource constraints and less on other parts of the generated SSRS. A detailed comparison between realism self-assessment and human expert evaluations is provided in a subsequent section.

Finally, the integration of self-refinement logic directly into the SSRS generation prompt yielded mixed outcomes. Its introduction in Iteration 6 coincided with a sharp drop in DoR scores, but this cannot be solely attributed to the self-refinement, as the DoR self-assessment itself also was changed to be more critical at this point. In Iteration 7, the self-refinement was adjusted to focus specifically on resource-constraints and their alignment with other requirements, which was a frequently identified weakness highlighted in earlier realism self-assessments. This change led to a substantial improvement in DoR scores. While causal attribution on the effects of this approach remains uncertain, these findings suggest that targeted self-refinement mechanisms may help address specific recurring quality weaknesses in LLM-generated content.

11.1.4. Chain-of-Thought Pattern

The Chain-of-Thought pattern was first introduced into the SSRS generation prompt in Iteration 5 with the intention of improving coherence and realism by prompting the model to reason step-by-step through each section of the SSRS template. This addition had no direct measurable effect on DoR scores. In Iteration 7, the Chain-of-Thought

pattern was combined with the targeted self-refinement instructions described in the previous section. This combination contributed to a notable improvement in DoR scores. However, in subsequent iterations, DoR scores fluctuated, both decreasing and increasing significantly, despite continued use of Chain-of-Thought. Based on these mixed results the effectiveness of Chain-of-Thought can not be assessed clearly for its effect on the generation of SSRS.

The limited effectiveness of Chain-of-Thought in this context may be attributed to the nature of SSRS generation task itself, because it is not easily decomposable into independent sub-steps. All template elements are linked together and must be developed in logical coherence with one another. This inherent interdependence likely reduces the utility of sequential reasoning strategies, making Chain-of-Thought a less suitable fit for this type of generative task and potentially also results in rather negative than positive effects on SSRS quality.

11.1.5. General Prompt Design Insights

Two key insights emerged during the iterative prompt refinement process. First, instruction clarity and conciseness is essential. Early vague formulations such as the generic cue to ”ensure that the scenario is distinct” yielded a noticeable improvement over having no diversity instruction at all, but also resulted in the highest overall variability in similarity scores across all iterations. Since it is not explicitly stated what defines a “distinct scenario”, the model is forced to interpret this instruction on its own, and due to the inherent randomness in LLM outputs, this interpretation can vary significantly across the 30 generated SSRSs, as reflected in the high score variability. This highlights a key lesson: prompts must be formulated so the model is left with no room for interpretation. Everything that is expected but also everything that is undesired must be precisely defined for the LLM. This is also reflect in the data. In the last iteration of the process a precise set of instructions was given to the LLM of how to generate diverse SSRSs, which resulted in the by far best average similarity score out of all iterations.

Second, the results highlighted that prompt overload can significantly diminish output quality. From iteration 3 to 7 there is a continuous worsening of semantic similarity scores, as the prompt grew significantly in size and complexity. While the results can not solely be attributed to the size and complexity of the prompt it is worth mentioning that the prompt of the last iteration contains 31% less words in the task instructions for generating an SSRS compared to the largest prompt. When comparing the data it gets evident that the shorter prompt of the last iteration produced SSRS with significantly better DoR and semantic similarity scores than the largest prompt used in iteration 8.

11.2. LLM Self-Assessment and Expert Judgments

The previous section analysed the general effectiveness and limitations of the employed prompting techniques primarily through the effects on the defined evaluation metrics. To more robustly assess the capabilities of LLM self-assessment, this section focuses

Template Element	Expert Realism Rating	Flagged by LLM	Match Type
Resource Constraints	22.6%	10/10	Match
User Base Characteristics	56.7%	0/10	Missed
Scalability	58.5%	9/10	Match
Reliability	64.2%	3/10	Match
Performance	67.9%	3/10	Match
Technical Constraints	69.8%	1/10	Partial
Authentication	70.4%	0/10	Missed
Integration Needs	71.7%	4/10	Match

Table 11.1.: Overlap between expert realism ratings and realism issues identified by the LLM during self-assessments.

on a direct comparison between the model's DoR self-assessments and domain-expert judgments. The goal of the following discussion is to examine the alignment between LLM-detected issues and expert-identified realism flaws, uncover potential blind spots in the model's self-assessment, and derive insights into when and how self-assessment can be a reliable mechanism in LLM-based generation workflows.

Table 11.1 presents a comparative analysis between the expert realism ratings and the LLMs self-assessment for the eight least realistic template elements in the expert study. For each element, the table reports the proportion of expert reviewers who rated it as "realistic," the number of times the LLM flagged that element as containing a realism issue across the same ten SSRSs evaluated in the expert study, and a qualitative match classification (Match, Partial, Missed) indicating alignment between expert and LLM judgments.

The "Flagged by LLM" values were derived by manually analysing the LLMs realism self-assessments for the same ten SSRSs rated by experts. For each self-assessment, any critique referencing a specific template element was recorded, and counts were aggregated across the dataset. It is important to note that the LLM often identified issues involving interdependencies, such as "Scalability and Budget Alignment" or "Reliability vs. Development Resources," rather than critiquing individual template elements in isolation.

The table reveals several notable patterns. First, for the template element of Resource Constraints with the lowest expert realism rating at 22.6%, the LLM also consistently flagged issues in all 10 SSRSs, indicating a complete match. A similar high alignment was observed for Scalability (58.5%), flagged in 9 out of 10 SSRSs. For Reliability, Performance, and Integration Needs, rated as realistic by 64.2% to 71.7% of experts, the LLM also flagged issues in 3 to 4 SSRSs, suggesting a generally consistent alignment between LLM and experts. However, for Technical Constraints, the LLM flagged an issue only for a single SSRS, while expert ratings suggest broader concerns with only 69.8% of experts marking it as realistic. In contrast, for the two elements of User Base Characteristics and Authentication the LLM failed to flag any issues at all, despite expert

realism scores of 56.7% and 70.4% respectively, indicating clear misses. These results suggest that while the LLM can detect certain types of realism issues effectively, it also exhibits blind spots.

As described there is a clear alignment between expert ratings and LLM self-assessments for the template element of Resource Constraints. While this may initially appear to demonstrate the LLMs ability in detecting feasibility issues related to budget, team size, or project timelines, a closer examination suggests a different interpretation. Unlike domain-specific features, resource-related aspects can not be considered an inherent properties of a software system but are solely depended on the context in which the software system should be developed. For example a given software system could be implemented under the constraints of a high budget, a large team size but a small timeline, or with a small team, a small budget and a long timeline. Additionally, it is plausible that information on the resource constraints of real-world software systems is rarely available in publicly accessible sources. Consequently, such data is likely to be sparsely reflected in the LLMs training data, limiting the model's ability to draw on accurate or contextually grounded examples when generating these aspects for a given SSRS. As a result, the LLM is likely to rely on assumptions or estimations when generating the resource constraints. These assumptions mostly lean towards overly optimistic values, as reflected in the expert evaluations, where budget and timeline estimates were frequently described as "too low" and team sizes as "too large". However, this issue does not only affect SSRS generation but also the DoR self-assessment as clearly displayed in the data, as the LLM identified realism issues with resource constraints for every single SSRS. With the before mentioned data-scarcity in mind this also leads to the likely explanation that because of missing reference data the LLM is not well equipped to assess whether the presented resource constraints are realistic. Under these conditions of limited reference data the LLM is more prone to hallucination, generating critique or realism feedback that appears plausible but is not grounded in actual domain knowledge or reference data. This may explain why the model systematically flags resource-related issues as it attempts to compensate for informational gaps by producing assessments that mimic critical evaluation, despite lacking the necessary context to substantiate them. This leads to an important insight for self-assessment approaches. When presenting an LLM with domain-specific content to assess, it should be evaluated, whether the LLM is likely to possess sufficient underlying knowledge, based on its training data, to give an accurate judgement.

In contrast to the consistent, but potentially inaccurate critique of resource constraints, the LLM failed to flag any issues for the template elements *User Base Characteristics* and *Authentication*, despite both ranking among the least realistic elements in the expert study. This divergence reveals an important asymmetry in the model's self-assessment behaviour. While resource-related aspects were frequently critiqued, even in the likely absence of large training data, elements such as user base or authentication received no critique at all under similarly uncertain conditions. A plausible explanation is again rooted in data limitations: for authentication, detailed and domain-specific configurations are unlikely to be publicly shared due to security concerns, reducing their presence in the training data. For user base characteristics, publicly available figures,

if shared, are possibly rather vague, overgeneralised, or inflated for marketing purposes, further degrading the reliability of any learned reference data. Thus, while both areas likely suffer from insufficient or skewed training data, the model displays a completely different behaviour, compared to the resource constraints, by not identifying a single realism issue across all ten SSRS.

The contrasting behaviour observed between the LLMs frequent critique of resource constraints and its complete omission of issues in user base characteristics and authentication likely stems from fundamental differences in how these elements relate to domain knowledge. Resource constraints are largely domain-independent as discussed before, therefore the LLM relies on internal consistency within the SSRS itself instead of assessing alignment with domain specific data. For example the identified resource constraints were almost exclusively criticised in conjunction with other requirements and not in isolation. This allows the LLM to detect possible realism issues without requiring any domain-specific knowledge.

In contrast, user base characteristics and authentication mechanisms are domain-specific data. For example software systems in the finance domain are likely to have stricter authentication requirements than an e-commerce software system, which in turn is likely to have a larger user base than a financial system. Assessing the realism of these elements thus requires external reference data and can not solely be assessed by internal comparison with other requirements. As probably only limited reference data on these elements is available in the LLMs training data, and because their realism cannot be meaningfully evaluated through internal consistency checks alone, the model appears to systematically avoid critiquing them. However, this remains a speculative hypothesis and cannot be verified due to the black-box nature of ChatGPT.

This observation highlights an important consideration for the design and interpretation of LLM-based self-assessments: one should be aware that the model's ability to identify issues depends on whether an element can be evaluated through internal consistency or requires external reference knowledge. In cases where external references are necessary, the absence of identified issues should not be taken as evidence of correctness. Instead, it may simply reflect the model's limited knowledge. Recognising this limitation is crucial when using self-assessment outputs to guide validation or prompt iteration.

Having examined how LLM self-assessments align with expert evaluations for the individual template elements, it is also important to consider the SSRS-Gen process as a whole. The following section examines the overall expert realism ratings in conjunction with the comments they provided for justification. Additionally for full transparency, changes in terminology during the execution of the SSRS-Gen process are discussed.

11.3. Review of the SSRS-Gen Process

Following the analysis of individual prompting strategies and the comparison between LLM self-assessments and expert evaluations, this section provides a broader review of the SSRS-Gen process as a whole. It reflects on both the perceived quality of the generated outputs. particularly their realism as judged by human experts, and on foun-

dational aspects of the prompt design itself. Special attention is given to the terminology used during generation, especially the consistent use of the term ”scenario” rather than ”SSRS”, which may have subtly influenced the model's interpretation of the task.

11.3.1. Output Realism and Expert Perception

An important dimension in evaluating the SSRS-Gen process concerns the quality of the generated outputs, particularly their degree of realism. Based on the expert study results, the overall realism of the generated SSRSs can be considered relatively high. A majority of participants rated the scenarios as either “somewhat realistic” (51%) or even “very realistic” (15.7%). However, deeper analysis of the received expert comments reveals a critical insight: experts who engaged in more thorough assessments consistently identified significant realism flaws, particularly in areas such as resource feasibility, authentication, or user base estimates. To give a concrete example, 70% of experts rated the section of authentication as realistic, while the other 30% of experts, as their comments suggest, did a more in-depth analysis and revealed important realism issues. This introduces a key cautionary insight: the confident tone and surface-level plausibility of LLM-generated outputs may create an illusion of realism that can mislead even experienced reviewers unless they actually perform an in-depth analysis of the generated content. This gets even more evident when inspecting the comments these participants wrote. Experts who rated their presented SSRS as overall “very artificial” or “somewhat artificial” frequently highlighted realism flaws that appear both reasonable and grounded in domain expertise. These comments pointed to a range of specific issues, including infeasible budget and timeline estimates, vague or unrealistic authentication mechanisms, inflated or unsupported user base assumptions, and inconsistencies in described system architectures. For instance, several participants noted that proposed development budgets would barely cover personnel costs, or that suggested timelines were incompatible with the described scope and integration needs. These observations suggest that, despite high overall realism ratings, deeper scrutiny reveals a pattern of recurrent and substantive issues. Thus, the LLM-generated SSRSs, while often convincing on first glance, may fail to withstand detailed professional evaluation. This insight underscores that LLMs are not yet fully capable of replacing the in-depth domain knowledge of experienced human experts, particularly when it comes to identifying nuanced or context-specific realism flaws in generated artefacts like SSRS. This is especially important in areas such as software security, where real-world data for topics such as authentication is likely to be rarely available in public sources and thus also likely under-represented in the model's training data. A summary of this insight is highlighted in the following warning box.

⚠ Caution: Plausible and Confident Outputs Can Mislead Human Experts

LLM-generated SSRSs can appear realistic at first glance, yet still contain critical flaws. The confident tone and surface-level plausibility of LLM outputs can cause even experienced experts to overlook issues unless the content is examined in detail.

In addition to evaluating the realism and expert assessments of the generated SSRSs, it is also important to reflect on the terminology used throughout the process, as this may have subtly influenced how the model interpreted the generation task.

11.3.2. Terminology During Process Execution

An important consideration in the SSRS-Gen process concerns the evolution of terminology and conceptual framing during the early stages of the project. At the outset, the artefacts targeted for generation were referred to as "evaluation scenarios," rather than SSRS. This naming reflected the preliminary understanding of the task, as well as the absence of a formalised artefact definition at the start of the thesis.

As the process matured, it became clear that the desired outputs aligned closely with system requirement specifications, resulting in the shift in terminology to SSRS. Despite this change in terminology, all ten iterations used the word "scenario" when instructing the model. This persistent use of the term "scenario" raises the question of whether the model may have interpreted the task differently, potentially generating outputs that are more narrative or abstract in nature, rather than structured and specification-like as intended for an SSRS. However, this risk is likely mitigated by the use of the SSRS template throughout all iterations. The template functioned as a structural and semantic foundation, consistently guiding the LLM to instantiate a fixed set of requirement categories regardless of the surrounding prompt wording. Thus, while the term "scenario" may have introduced some ambiguity, the template itself constrained the model's generative behaviour to remain within the intended SSRS format.

Still, future work could further investigate the sensitivity of generation outcomes to such terminology. For example, a direct comparison between prompts using the term "scenario" versus "SSRS" could help evaluate whether naming conventions impact model behaviour or the perceived realism of generated outputs.

11.4. Limitations of the Study

While the findings of this work offer valuable insights into the capabilities and challenges of using LLMs for SSRS generation and evaluation, several limitations must be acknowledged. These span technical constraints related to the used LLM, methodological boundaries of the prompt engineering process, and the design of the expert study used for validation. The following sections detail these constraints and their implications for the generalisability and robustness of the presented findings.

11.4.1. LLM Model Limitations

The SSRS-Gen process relied exclusively on ChatGPT-4o, accessed through the public web interface. This imposes several structural limitations on transparency and interpretability. As a proprietary black-box model, ChatGPT offers no visibility into its internal architecture, training data composition, or parameter configuration. Consequently, users cannot assess the representativeness, quality, or domain coverage of the model's training data, which is a significant concern for domain-specific generation tasks like SSRS-Gen. This lack of transparency is especially problematic for SSRS generation tasks, where it is unclear whether the model for example has sufficient training data on real-world authentication practices to generate plausible specifications and subsequently also assess their realism reliably.

Additionally, using the web-based interface of ChatGPT does not allow for control over any generation settings, such as how much variation or randomness the model introduces during output creation. Although the prompting procedure was equivalent for all SSRS, the model's inherent variability may have influenced individual outputs. Since identical prompts may still yield different responses, this limits the reproducibility of results and complicates their evaluation.

Another limitation arises from the evolving nature of the ChatGPT platform itself. The underlying specific model version may be updated by OpenAI at any time without explicit notice to users. As a result, it is highly likely that different versions of the model were used during the course of the ten SSRS-Gen iterations, even though the interface remained the same. For example, the version employed in Iteration 1 was almost certainly not identical to that used in Iteration 10. This lack of version transparency introduces further uncertainty into the evaluation process, as observed changes in output quality may be partially attributable to model updates rather than the applied prompting strategies alone.

11.4.2. Prompt Engineering Constraints

The prompt engineering process in SSRS-Gen followed a manual approach, where prompt modifications were guided iteratively by observation of the metric data and intuition rather than a fully systematic approach. As described in section 6.6, the SSRS-Gen process employed an iterative refinement strategy guided by observed weaknesses and informed by the prompt engineering literature. This approach was chosen over a systematic or combinatorial method due to the sheer complexity of the design space: not only are there numerous prompt patterns to consider, but each pattern's effect can vary significantly depending on its specific formulation and contextual integration, for example which expert role is used in a persona prompt. As a result, exhaustively exploring all pattern combinations and their implementations is practically infeasible. While this selective and adaptive strategy is well-justified for exploratory research, it limits the ability to isolate the individual contribution of each pattern or formulation. Consequently, conclusions about the effectiveness of specific prompting strategies must be viewed in the context of the trade-off made between achieving precise, isolated measurements of

each prompt pattern and maintaining a practical, manageable experimental process.

Additionally, the prompts were evaluated solely within the SSRS generation context and across a fixed set of industry domains. It remains unclear whether the prompts developed in this process would generalise effectively to other types of system specification tasks or different domain contexts. While it is reasonable to assume that the process could be adapted to other domains by replacing the content of the template and adjusting the prompts accordingly, this assumption currently lacks empirical support. The generalisability of the designed prompts beyond the scope of SSRS-Gen thus remains limited and would require further experimentation and validation in alternative contexts.

Another limitation concerns the use of fine-grained scoring scales in LLM-based self-assessment prompts. In this work, a decimal scale from 0.00 to 1.00 was employed to encourage nuanced realism evaluations. However, recent findings suggest that large language models may struggle to consistently and meaningfully utilise high-resolution scales. Specifically, Liu et al. [SAS24] demonstrate that when LLMs are prompted to score on a 1–100 scale, the models tend to exhibit round-number biases, overusing values like 90 and 95 while neglecting much of the lower range. This leads to skewed and less informative distributions, reducing the practical benefits of higher granularity. This behaviour was also observed in the execution of the process, as ChatGPT often used 0.05-steps when providing a DoR score. This issue was not known during the development of this process. Thus attempts were made to encourage the model to use finer 0.01-step increments instead of 0.05, without being aware of the potential drawbacks. This decision was instead based on the assumption that greater granularity would improve scoring accuracy. However, this intervention may have further amplified inconsistencies in the DoR self-assessment. [SAS24].

11.4.3. Expert Study Design

A key limitation of the expert study relates to participant sampling and qualification verification. The study employed purposive sampling with additional snowball recruitment, deliberately targeting individuals with relevant professional backgrounds. While this approach aimed to attract qualified experts, it does not allow for full verification of participant expertise. Although several screening questions were used, including filters for professional software experience, self-identification as a domain and/or technical expert, years of experience, and domain-specific confidence, these relied on self-reporting and could not be independently verified. Consequently, while the participant pool likely included mostly qualified experts, the possibility remains that some respondents lacked the depth of expertise required for a rigorous realism evaluation. This limitation introduces some uncertainty regarding the consistency and reliability of the expert judgments, though the study design aimed to minimise this risk through targeted recruitment and domain and expertise filtering.

Another limitation arises from the selection of SSRSs evaluated in the expert study. Out of the 30 SSRSs generated in the final iteration, only one per industry domain was randomly selected for expert review, so a sample size of ten SSRS in total. Although

the selection was randomised by domain, the reduced sample size introduces the risk that the evaluated SSRSs may not fully reflect the quality of all other SSRS within the entire set. However, this risk is considered rather minimal, as realism self-assessments for all SSRSs within the last iteration showed a high degree of consistency, suggesting limited variation in output quality. Nonetheless, future studies could be employed for evaluating the full set of generated SSRSs.

A further limitation concerns potential fatigue effects during the expert evaluation process. The SSRS presented for review are relatively long and complex, requiring sustained attention across multiple detailed sections. This may have introduced cognitive fatigue, particularly for participants attempting to complete the entire assessment in a single session. Evidence of this effect is reflected in the decreasing response count across sections: in all 60 questionnaires the first section was completed, but only 53 cases completed all four sections. Additionally in just 51 cases an answer was provided for the overall realism assessment at the end. This drop-off suggests that some participants may have disengaged toward the end of the evaluation, which could affect the consistency and depth of their judgments. Future studies might mitigate this risk by distributing the assessment over multiple sessions, limiting the scope per session, or implementing engagement checks.

Another limitation arises from the potential influence of confirmation bias in expert evaluations. Because participants were informed that the SSRSs were generated by an LLM, their judgments may have been unconsciously shaped by pre-existing attitudes toward AI-generated content. Given the ongoing public discourse around AI and LLMs, it is plausible that participants held differing preconceptions about the reliability of LLM-generated content, which may have influenced their judgments. Professionals with a more positive outlook on the capabilities of LLMs may have evaluated the outputs more favourably, while experts with a rather sceptic view on LLMs may have applied a higher level of scrutiny, regardless of the actual quality.

These limitations not only constrain the current findings but also highlight important opportunities for refinement and expansion, which are discussed in the following section on future work.

11.5. Research Questions Revisited

This section revisits the three central research questions introduced at the beginning of this thesis, integrating findings from the SSRS-Gen process implementation, the LLM self-assessments, and the follow-up expert study.

RQ1: *How can large language models be systematically applied to generate high-quality synthetic system requirement specifications in contexts where real-world system requirement specifications are unavailable?*

To address this question, a structured generation process was developed based on a targeted literature review on prompt engineering techniques and evaluation metrics.

The process leverages zero-shot prompting in combination with prompt patterns such as the template pattern, persona pattern, and structured reasoning techniques. Additionally, self-assessment prompts were integrated to support iterative refinement. Together, these components enabled the generation of domain-specific SSRSs in the absence of real-world examples or evaluation by human experts. The process was designed to be repeatable, scalable, and adaptable across different industry domains and application contexts, demonstrating that LLMs can be applied systematically to produce artefacts approximating the structure and content of real SyRSs.

RQ2: *How can the quality of synthetic system requirement specifications be defined and measured in the absence of human experts?*

To enable quality assessment without relying on expert review, the thesis defined a set of evaluation metrics, based on the process requirements of *R2-Comparability*, *R1-Realism* and *R3-Diversity*, as defined in section 4.1. Each process requirement was addressed by a custom metric: completeness for structural adherence to the SSRS template, degree of realism for the plausibility of generated content, and semantic similarity for output diversity. The metrics of completeness and DoR were measured using LLM self-assessment prompting the model to evaluate its own outputs. While this approach does not replace expert judgment, it provided a scalable and automated means of identifying both structural adherence and possible realism issues within the generated artefacts.

RQ3: *How do human experts evaluate the quality of the LLM-generated synthetic system requirement specifications?*

To evaluate how human experts perceive the quality of the generated SSRSs, a questionnaire expert study was conducted covering a sample of 10 out of 30 SSRS generated in the last iteration of the process execution, covering each of the ten industry domains once. Overall, half of the participants rated their presented SSRS as ”somewhat realistic”, and nearly 16 percent of participants even as ”very realistic”. These results indicate that the SSRSs were often convincing and perceived as realistic, at least at a surface level. However, qualitative feedback revealed that this impression did not always hold under closer scrutiny. Experts identified recurring weaknesses such as vague phrasing, overly generic content, unrealistic assumptions, and inconsistencies between requirement sections. These issues often became apparent only when reviewers examined the outputs critically and in detail. Notably, the confident tone and fluent structure of the SSRSs occasionally led to the overestimation of their quality, suggesting that LLMs can produce outputs that appear credible but contain subtle or significant flaws. These findings highlight two key takeaways. First, that review by experienced human experts remains essential and can currently not be replaced by automated assessment alone. Second, that the SSRS-Gen process can be further improved based on the concrete weaknesses observed in the study. This latter point directly informs the directions for future work presented in the next section.

11.6. Future Work

While the SSRS-Gen process presented in this thesis demonstrates promising results in generating realistic SSRSs using LLMs, there are still several directions for further research and development. These include expanding the scope of expert evaluations to cover all 30 SSRS of the last iteration, exploring the adaptability of the SSRS-Gen framework to new domains and system specification contexts, and enhancing the generation and self-assessment mechanisms through refined prompting strategies. Together, these directions aim to strengthen the reliability, generalisability, and practical utility of the proposed SSRS-Gen process.

11.6.1. Full-Scale Expert Evaluation

A key direction for future work involves conducting a full-scale expert evaluation encompassing all 30 SSRSs generated in the final iteration. In the present study, only one SSRS per domain was randomly selected for expert review, limiting the generalisability of the findings across the complete dataset. While the same target group of domain and technical experts would likely be appropriate, revisiting the evaluation design could help reduce fatigue effects observed in the current setup. A broader evaluation would not only enable more statistically robust findings but also offer the opportunity to encourage more in-depth, critical reviews from all participants. This could help surface nuanced realism issues across the entire SSRS set and provide deeper insights into the current limitations of LLM-generated SSRSs.

11.6.2. Generalisation to Other Task Contexts

While the SSRS-Gen process in this work focused specifically on generating SSRSs across a fixed set of industry domains and for the application context of the SecuRe recommender system, the SSRS-Gen process offers potential for broader application. The approach could be adapted by modifying the content of the SSRS template according to the specific needs for a given evaluation and testing task. For practitioners interested in adapting this process, it is also recommended to begin with the final iteration prompts developed in this thesis and modify them according to their specific goals, rather than restarting from a minimal prompt and re-running the full iterative design loop. The final prompt versions already integrate tested strategies such as expert persona, targeted self-assessment, and detailed task instructions, and thus should offer a robust starting point for extension or domain adaptation.

While the SSRS-Gen process offers strong potential to be adaptable, this remains a hypothesis. No tests were conducted outside the defined context, and the generalisability of the process must therefore be validated empirically. Given the realism issues uncovered through expert evaluation in this work, any future use of the process in a new context should still include external review by qualified experts to verify plausibility and alignment with real-world expectations.

11.6.3. Enhancing Prompts and Self-Assessment

The expert evaluation revealed several shortcomings in the realism of certain SSRS sections. Additionally, the direct comparison between LLM-generated self-assessments and expert feedback uncovered multiple blind spots, where the model either overlooked unrealistic elements or critiqued aspects it was poorly equipped to evaluate. While the current self-assessment prompts represent a promising step toward output validation, they require further refinement to improve diagnostic accuracy and alignment with human expert standards. Future research should explore the further refinement of the SSRS-Gen process focusing on both the SSRS generation as well as the realism self-assessment. Although the overarching goal remains to enable reliable SSRS generation without human intervention, expert review should still be employed in future iterations to verify both the realism and overall quality of model outputs until the self-assessment process and the SSRS generation quality itself are sufficiently robust for practical deployment.

12. Conclusion

This concluding chapter synthesizes the findings and insights developed throughout the thesis, reflecting briefly on the core research questions and their outcomes. It emphasizes key implications of the presented work and points toward future research opportunities, aiming to provide a clear, final perspective on the contributions and limitations of using large language models in structured requirements engineering tasks.

12.1. Summary of Contributions

This thesis investigated the application of large language models (LLMs) for generating realistic, domain-specific Synthetic System Requirement Specifications (SSRSs) in contexts where access to real-world system requirement specifications (SyRS) or domain experts is limited. Addressing this challenge, the work introduced and validated the SSRS-Gen process: a structured, repeatable methodology that leverages zero-shot prompting, prompt engineering techniques, and self-assessment strategies to generate and evaluate SSRSs across multiple industry domains.

To support this process, an SSRS template was developed, grounded in the standard ISO/IEC/IEEE 29148:2018 [ISO18] and extended with authentication-relevant attributes tailored to the SecuRe recommender system, which served as a concrete application context. The methodology incorporated targeted literature reviews to identify effective prompting strategies and evaluation metrics, which were operationalized in a four-phase process comprising generation, completeness assessment, realism evaluation, and semantic similarity measurement.

The process was implemented using ChatGPT as a black-box LLM and iteratively refined across ten prompt iterations, generating 300 SSRSs in total. Output quality was evaluated both through LLM self-assessments and a follow-up expert questionnaire study. The results indicate that with scientifically evaluated prompting techniques, LLMs are able to generate SSRSs that are structurally consistent and contextually plausible. Additionally, the use of LLM self-assessment showed potential as a scalable complement to human expert evaluation, but it is also subject to limitations, which must be carefully considered .

Collectively, this work proposes a structured approach to the generation of SSRSs, with potential applicability in evaluation and testing tasks in software engineering domains where access to real-world SyRS or human experts is limited. It lays a conceptual foundation for future investigations into LLM-assisted requirements engineering beyond the specific context explored in this thesis.

12.2. Key Insights and Implications

This thesis offers several key insights into the use of LLMs for generating structured artefacts in domains characterized by limited access to real-world data and expert input. While the initial motivation arose from the specific needs of a constraint-based software security recommender system, the findings carry broader relevance for evaluation-driven workflows in software engineering and beyond.

A central insight is that zero-shot prompting, when supported by carefully selected prompt patterns, such as the Template and Persona patterns, can guide LLMs toward producing SSRS that are structurally complete and contextually plausible. The integration of structured reasoning and self-assessment techniques further improved the quality and consistency of outputs, particularly when applied in iterative refinement loops. These findings suggest that prompt engineering should be employed as a fundamental component in shaping LLM behaviour for structured and domain-specific text generation tasks.

Another important implication is the demonstrated potential of LLM self-assessment techniques to support initial quality evaluation of generated SSRS. While not being a replacement for human expertise, especially in complex or safety-critical domains, these techniques offer a scalable way to flag issues in settings where expert input is limited. The study showed that LLM-based assessments aligned with expert judgments in many cases, particularly when issues were detectable through general reasoning or internal consistency. However, the model also exhibited blind spots, failing to identify certain unrealistic assumptions and occasionally overestimating the plausibility of ambiguous or oversimplified content. These discrepancies highlight the uneven reliability of LLMs in evaluative roles and reinforce both the need for hybrid evaluation strategies by combining automated feedback with targeted expert review and the opportunity to further improve and refine the self-assessment approach introduced in this thesis.

Together, these insights point to a promising but cautious path forward for the integration of LLMs into requirements engineering and evaluation tasks. Their effectiveness depends not only on model capabilities, but on the surrounding process architecture including prompt design, refinement, and output quality assessment, all of which must be systematically engineered to ensure reliability and usefulness.

12.3. Reflective Summary of Research Questions

This thesis was guided by three central research questions addressing the generation, evaluation, and expert perception of Synthetic System Requirement Specifications produced by an LLMs. The first question explored how LLMs can be systematically applied to generate high-quality SSRSs in the absence of real-world data. Through the development and iterative refinement of the SSRS-Gen process, which incorporated scientifically grounded prompt engineering techniques, a structured approach was proposed and tested within the context of the SecuRe recommender system and across ten distinct industry domains.

The second research question concerned how the quality of SSRSs can be assessed without relying on human experts. In response, a combination of LLM-based self-assessment techniques, automated evaluation, and custom metrics was introduced. These methods were effective in capturing structural completeness and semantic diversity, and provided a practical means of evaluating realism without relying on expert feedback. However, the findings also revealed important limitations, particularly in the consistency and sensitivity of realism assessments. This highlights both the continued necessity of expert involvement in evaluating nuanced quality aspects and the need for further refinement of self-assessment techniques.

Finally, the third question examined how domain experts perceive the outputs generated by the process. The expert evaluation revealed that overall SSRSs were viewed as realistic, validating the general plausibility of the approach. However, critical feedback also highlighted recurring weaknesses, such as generic phrasing, over-ambitious requirements, and occasional logical inconsistencies. This finding underscores the continued importance of human oversight in high-stakes applications.

Taken together, the findings provide an initial answer to each of the guiding questions, while also pointing to open challenges and opportunities for refinement in future work.

12.4. Concluding Remarks

This thesis explored the use of LLMs to generate Synthetic System Requirement Specifications in domains where access to real-world data and expert input is limited. By designing, implementing, and evaluating the SSRS-Gen process, it demonstrated that structured prompt engineering and self-assessment techniques can support the creation of realistic and diverse artefacts suitable for evaluation and testing applications.

While the results are promising, they also emphasize the limitations of current LLM-based methods, particularly in reliably judging realism and nuance without human oversight. These findings reinforce the importance of continuing to refine self-assessment techniques and for treating LLMs as supportive tools, rather than replacements for expert judgment.

Ultimately, this work contributes a foundation for further research into how LLMs can be thoughtfully embedded into requirements engineering workflows. As language models continue to evolve, the need for careful process design, transparency, and evaluation will remain critical to their responsible use in structured, high-stakes domains.

A. Appendix

You are a highly experienced Requirements Engineer and Business Analyst specializing in Telecommunications software systems. Your task is to generate a highly realistic, unique scenario addressing the challenges and objectives specific to the Telecommunications domain.

Key Instructions:
- Use the provided template to structure the scenario, ensuring all required sections are included and complete.
- Exclusions: Strictly avoid any references to authentication methods/patterns (e.g. simple password, Multi-Factor Authentication, biometrics, etc.) but include Authentication Conditions & Frequency (e.g., session expiration, sensitive actions).
- Diversity:
  - The generated scenario should be distinct from any previously generated scenarios.
  - Most importantly choose diverse system purposes and core features
  - Introduce variation in User base sizes, operational environment and geographic locations.
  - Vary Non-Functional Requirements and Constraints while adhering to domain standards and best-practices.
- Realism: Ensure the scenario represents best-practices and common requirements and constraints in the Telecommunications domain.

After drafting the scenario evaluate and refine the scenario. For each of the following two points, carefully assess the scenario, identify any issues, and make improvements before proceeding to the next point:
1. Verify allocated resources (budget, timeline, and team size) suffice to meet functional and non-functional requirements and any potential unforeseen events. Take into consideration how the non-functional requirements may impact resources and use a conservative rather than optimistic estimation including a large enough buffer for unforeseen events.
2. Compare the drafted scenario to any previously generated scenarios to ensure the new scenario is distinct from previous ones, reflecting unique system purposes, core features, user base sizes, operational environment and geographic locations

Deliver only the refined, domain-specific scenario that adheres to best practices and realistic objectives for Telecommunications software systems. Do not add any of your intermediate reasoning steps.

Template:

Figure A.1.: Final SSRS generation prompt incorporating persona assignment, structured task instructions and self-refinement steps combined with Chain-of-Thought prompting.

Verify whether the created scenario completely implements each point in the template. For each point and all sub-points in the template, list its name followed by <True | False> depending on whether that point was included in the scenario. After listing all points, provide a final combined answer (<True | False>), which should be True only if all individual points are True. If the final combined answer is False, briefly explain which points are missing or incomplete. Always end with: 'Final Completeness Assessment: <True | False>'

Figure A.2.: Completeness assessment prompt used in the final iteration.

You are a Senior Requirements Engineer and Business Analyst specializing in Telecommunications, with extensive experience evaluating the realism of business and technical requirements. Your expertise includes identifying gaps, risks, and potential issues in proposed solutions, aligning them with industry-specific goals, technical constraints, and best practices.

Your task is to assess whether the details within the scenario are realistic and logically aligned with the Telecommunications domain. Focus exclusively on the information provided in the scenario, without trying to identify any missing information.

Evaluation Criteria
1. Comprehensive Review: Assess all points and sub-points in the scenario, ensuring logical coherence and realistic interdependencies (e.g., resources vs. requirements, scalability vs. budget).
2. Scoring Method: Provide a decimal score between 0 and 1:
- 1.0: Entirely realistic and feasible.
- 0.0: Entirely unrealistic.
- Deduct points in steps of 0.01, scaled to the severity of the issue:
  - Minor (e.g., slightly underestimated resources, feasible but ambitious goals): 0.01–0.02.
  - Moderate (e.g., misaligned resources, tight timelines, scalability risks): 0.03–0.05.
  - Major (e.g., severe resource shortfall, infeasibility of core goals): 0.06–0.10.
3. Justify Deductions:
- List each unrealistic point with:
  - A brief description of the issue.
  - The exact deduction and reasoning based on its significance.

Output Structure:
- Summary of findings.
- Unrealistic points with justifications and deductions.
- End with the Final score: 'Final Realism Score: <Final Score>'.

Deliver a fair, balanced evaluation aligned with best practices in the Telecommunications domain.

Figure A.3.: Final Degree of Realism assessment prompt incorporating persona assignment and structured task instructions.

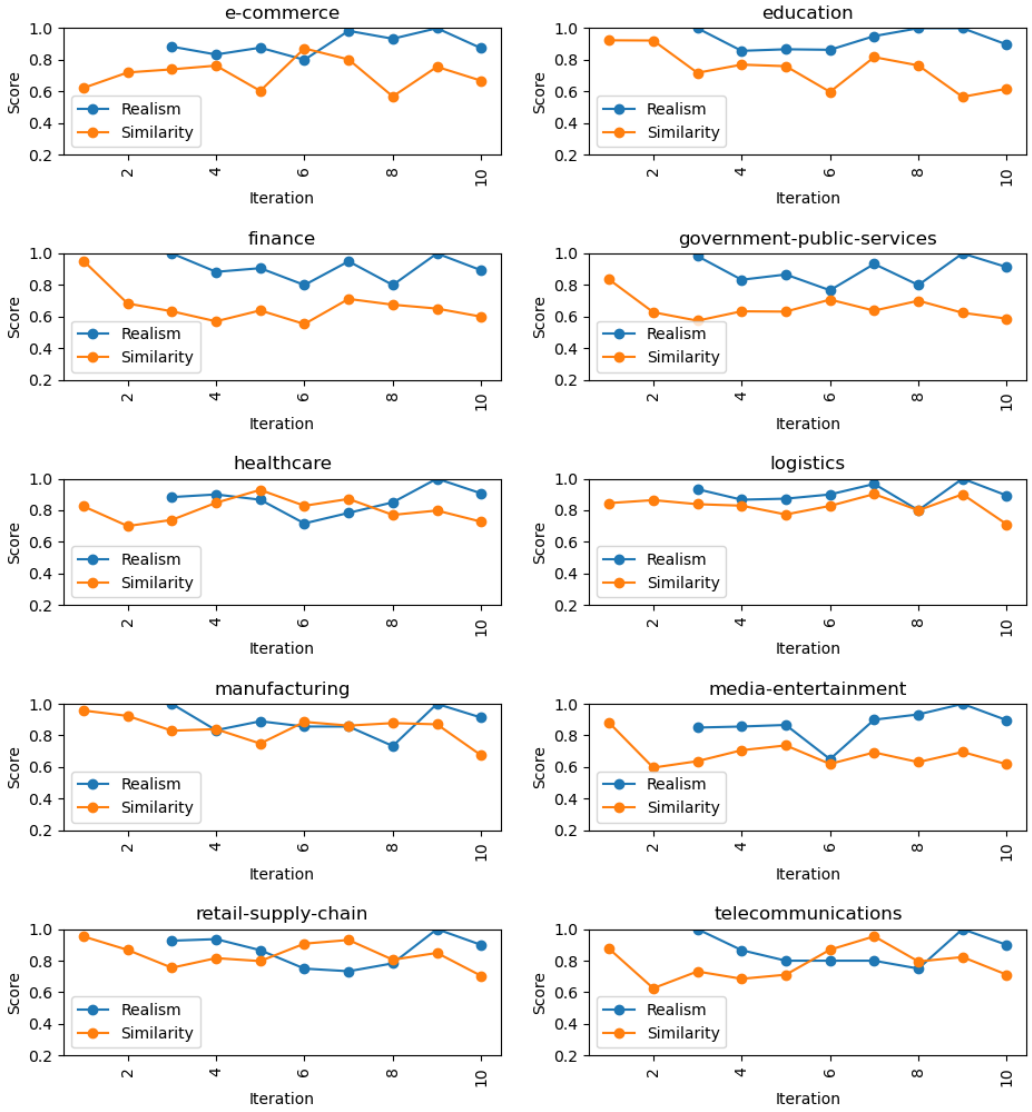

Figure A.4.: Similarity and realism scores across all iterations with one sub-plot for each domain.

Figure A.5.: Average similarity scores per industry domain across iterations.

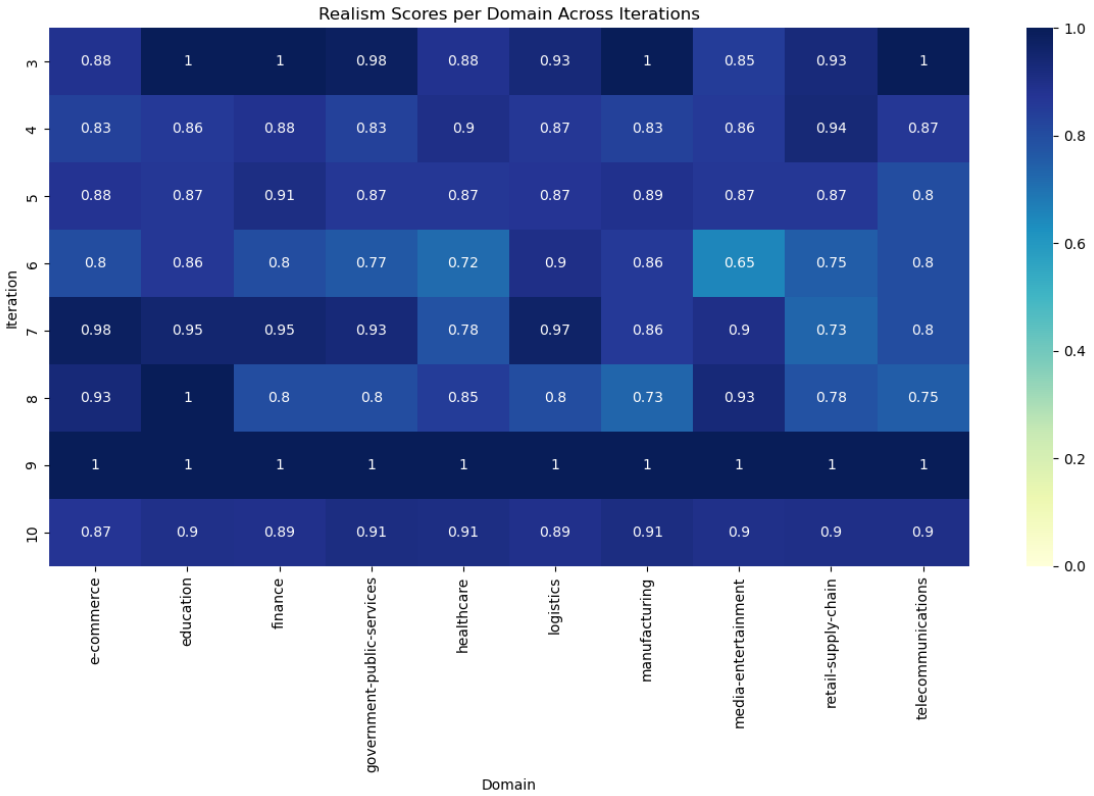

Figure A.6.: Average realism scores per industry domain across iterations.

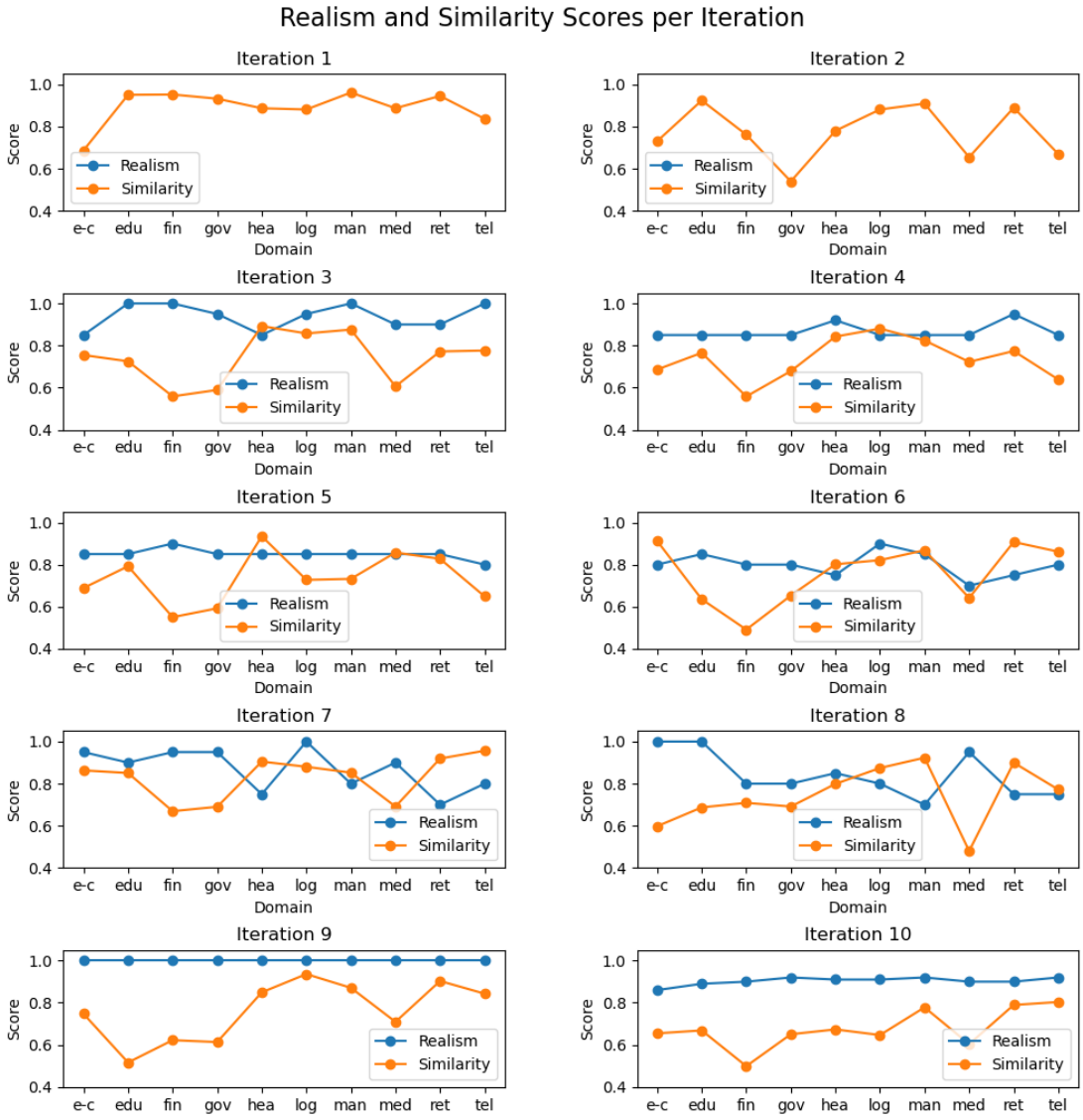

Figure A.7.: Subplots for each iteration displaying the realism and similarity scores of each IndDom.

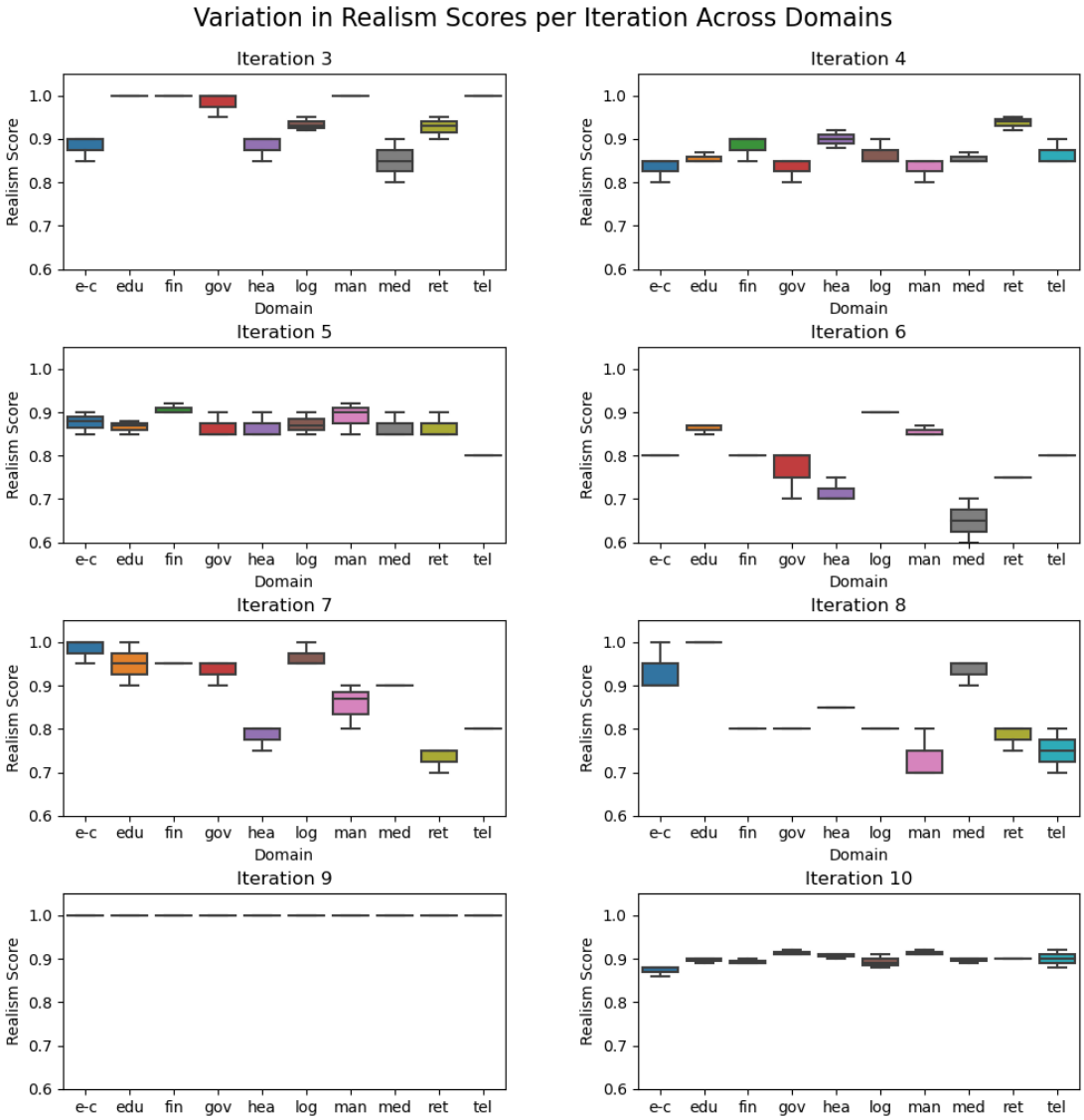

Figure A.8.: Variation in realism scores with one subplot for each iteration.

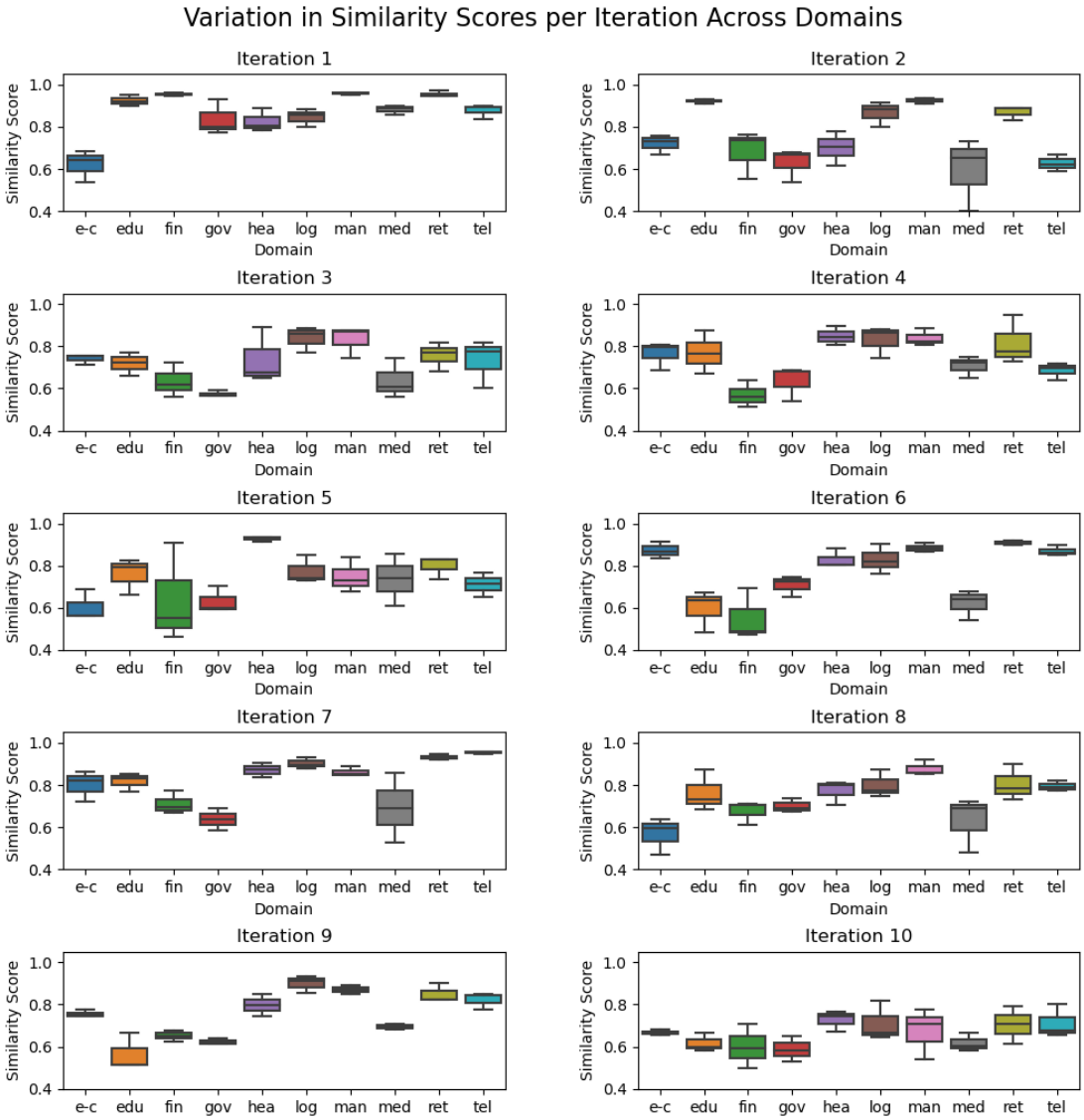

Figure A.9.: Variation in similarity scores with one subplot for each iteration.

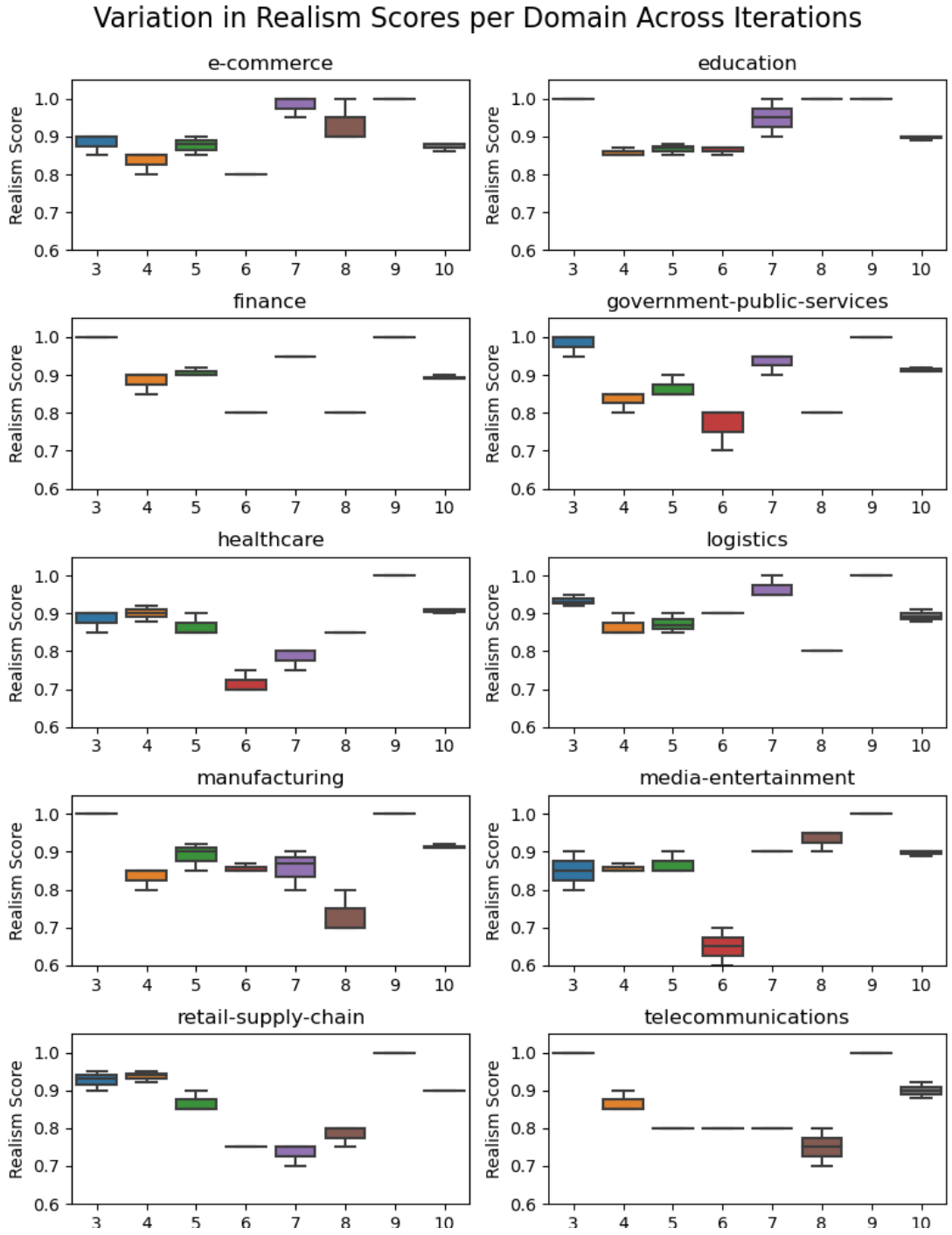

Figure A.10.: Variability of realism scores across iterations with one sub-plot for each domain.

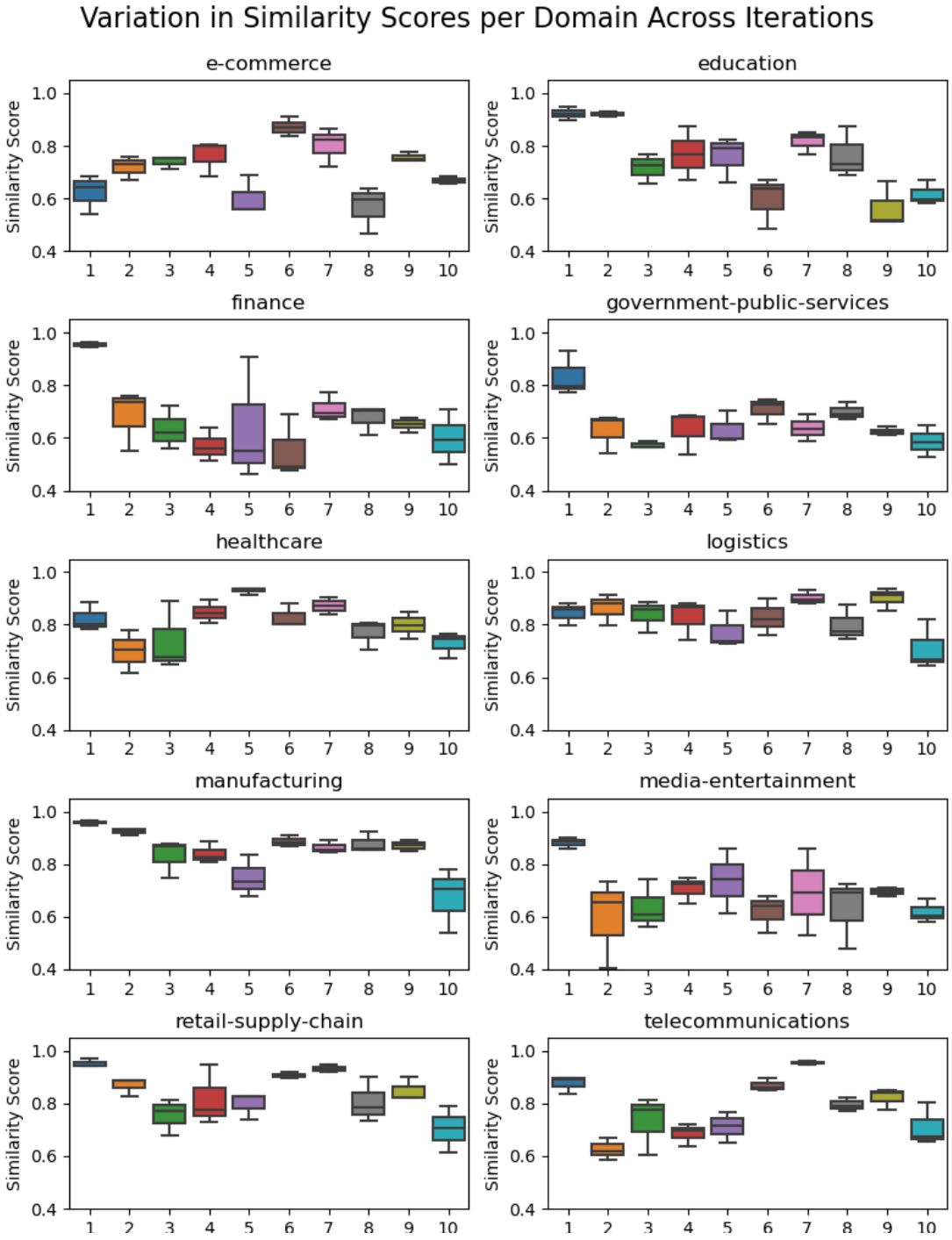

Figure A.11.: Variability of similarity scores across iterations with one sub-plot for each domain.

Bibliography

[Agg16] C. C. Aggarwal. *Recommender Systems: The Textbook*. 1st ed. 2016. Cham: Springer International Publishing, 2016. ISBN: 9783319296593 (cit. on pp. 1, 5, 6).

[AMM23] S. Arulmohan, M.-J. Meurs, and S. Mosser. "Extracting Domain Models from Textual Requirements in the Era of Large Language Models." In: *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. 2023, pp. 580–587. DOI: 10.1109/MODELS-C59198.2023.00096 (cit. on p. 77).

[Bas+25] L. Bass et al. *Engineering AI Systems: Architecture and DevOps Essentials*. Addison-Wesley Professional, 2025 (cit. on p. 9).

[BL05] S. Banerjee and A. Lavie. "METEOR: An automatic metric for MT evaluation with improved correlation with human judgments." In: *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*. 2005, pp. 65–72 (cit. on p. 28).

[Bro+20] T. B. Brown et al. *Language Models are Few-Shot Learners*. May 28, 2020. URL: http://arxiv.org/pdf/2005.14165v4 (cit. on pp. 10, 14).

[FB20] S. Furnell and M. Bishop. "Addressing cyber security skills: the spectrum, not the silo." In: *Computer fraud & security* 2020.2 (2020), pp. 6–11 (cit. on p. 6).

[Fel15] Felfernig, Alexander and Friedrich, Gerhard and Jannach, Dietmar and Zanker, Markus. "Constraint-Based Recommender Systems." In: *Recommender Systems Handbook*. Ed. by Ricci, Francesco and Rokach, Lior and Shapira, Bracha. Boston, MA: Springer US, 2015, pp. 161–190. ISBN: 978-1-4899-7637-6. DOI: 10.1007/978-1-4899-7637-6{\textunderscore}5 (cit. on pp. 1, 5, 6).

[FSG17] A. Ferrari, G. O. Spagnolo, and S. Gnesi. "PURE: A Dataset of Public Requirements Documents." In: *2017 IEEE 25th International Requirements Engineering Conference (RE)*. 2017, pp. 502–505. DOI: 10.1109/RE.2017.29 (cit. on pp. 1, 5, 7, 8).

[Gen+23] J. Geng et al. *A Survey of Confidence Estimation and Calibration in Large Language Models*. Nov. 14, 2023. URL: http://arxiv.org/pdf/2311.08298v2 (cit. on p. 12).

[Hou+23] B. Hou et al. "Decomposing uncertainty for large language models through input clarification ensembling." In: *arXiv preprint arXiv:2311.08718* (2023) (cit. on p. 13).

[Hu+23] M. Hu et al. *Uncertainty in Natural Language Processing: Sources, Quantification, and Applications*. June 5, 2023. URL: http://arxiv.org/pdf/2306.04459v1 (cit. on p. 13).

[Hua+24a] L. Huang et al. "A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions." In: *ACM Transactions on Information Systems* 39 (2024), p. 324. ISSN: 1046-8188. DOI: 10.1145/3703155. URL: http://arxiv.org/pdf/2311.05232v2 (cit. on pp. 9–11).

[Hua+24b] Y. Huang et al. *Calibrating Long-form Generations from Large Language Models*. Feb. 9, 2024. URL: http://arxiv.org/pdf/2402.06544v2 (cit. on p. 12).

[ISM19] T. Iqbal, N. Seyff, and D. Mendez. "Generating Requirements Out of Thin Air: Towards Automated Feature Identification for New Apps." In: *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*. 2019, pp. 193–199. DOI: 10.1109/REW.2019.00040 (cit. on pp. 73, 78).

[ISO18] ISO/IEC/IEEE. *ISO/IEC/IEEE 29148:2018 – Systems and software engineering – Life cycle processes – Requirements engineering*. International Standard. 2018. URL: https://www.iso.org/standard/72089.html (cit. on pp. 1, 16, 78, 95).

[Ji+23] Z. Ji et al. "Survey of Hallucination in Natural Language Generation." In: *ACM Computing Surveys* 55.12 (2023), pp. 1–38. ISSN: 0360-0300. DOI: 10.1145/3571730 (cit. on pp. 9–11).

[Kim+23] D.-K. Kim et al. "Assessment of ChatGPT's Proficiency in Software Development." In: *2023 Congress in Computer Science, Computer Engineering, & Applied Computing (CSCE)*. 2023, pp. 2637–2644. DOI: 10.1109/CSCE60160.2023.00421 (cit. on pp. 76, 78).

[Kos24] M. Kosinski. *What is black box artificial intelligence (AI)?* Oct. 2024. URL: https://www.ibm.com/think/topics/black-box-ai (visited on 05/27/2025) (cit. on p. 11).

[LHS24] D. Luitel, S. Hassani, and M. Sabetzadeh. "Improving requirements completeness: Automated assistance through large language models." In: *Requirements Engineering* 29.1 (2024), pp. 73–95 (cit. on p. 76).

[Li+24] D. Li et al. *From Generation to Judgment: Opportunities and Challenges of LLM-as-a-judge*. Nov. 25, 2024. URL: http://arxiv.org/pdf/2411.16594v3 (cit. on pp. 16, 27).

[Lia+24] X. Liang et al. *Internal Consistency and Self-Feedback in Large Language Models: A Survey*. July 19, 2024. URL: http://arxiv.org/pdf/2407.14507v3 (cit. on pp. 10, 11).

[Lin04] C.-Y. Lin. "Rouge: A package for automatic evaluation of summaries." In: *Text summarization branches out*. 2004, pp. 74–81 (cit. on p. 28).

[Mad+23] A. Madaan et al. *Self-Refine: Iterative Refinement with Self-Feedback*. Mar. 30, 2023. URL: http://arxiv.org/pdf/2303.17651v2 (cit. on p. 16).

[Mai+24] P. Maini et al. "LLM Dataset Inference: Did you train on my dataset?" In: *Advances in Neural Information Processing Systems*. Ed. by A. Globerson et al. Vol. 37. Curran Associates, Inc., 2024, pp. 124069–124092. URL: https://proceedings.neurips.cc/paper_files/paper/2024/file/e01519b47118e2f51aa643151350c905-Paper-Conference.pdf (cit. on p. 9).

[Mat17] R. Matulevičius. *Fundamentals of secure system modelling*. Springer, 2017 (cit. on p. 6).

[MLG23] P. Manakul, A. Liusie, and M. J. F. Gales. *SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models*. Mar. 15, 2023. URL: http://arxiv.org/pdf/2303.08896v3 (cit. on pp. 10, 11).

[Oxf23] Oxford University Press. *Realistic, adj., sense 1.a*. Oxford English Dictionary. Retrieved May 26, 2025. July 2023 (cit. on p. 18).

[Pap+02] K. Papineni et al. "Bleu: a method for automatic evaluation of machine translation." In: *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 2002, pp. 311–318 (cit. on p. 28).

[PMR24] J. Peer, Y. Mordecai, and Y. Reich. "NLP4ReF: Requirements Classification and Forecasting: From Model-Based Design to Large Language Models." In: *2024 IEEE Aerospace Conference*. 2024, pp. 1–16. DOI: 10.1109/AERO58975.2024.10521022 (cit. on pp. 77, 79).

[PR20] S. Panichella and M. Ruiz. "Requirements-Collector: Automating Requirements Specification from Elicitation Sessions and User Feedback." In: *2020 IEEE 28th International Requirements Engineering Conference (RE)*. 2020, pp. 404–407. DOI: 10.1109/RE48521.2020.00057 (cit. on pp. 74, 78).

[Qia+22] S. Qiao et al. *Reasoning with Language Model Prompting: A Survey*. Dec. 19, 2022. URL: http://arxiv.org/pdf/2212.09597v8 (cit. on p. 10).

[Rei+20] R. Rei et al. *COMET: A Neural Framework for MT Evaluation*. Sept. 18, 2020. URL: http://arxiv.org/pdf/2009.09025v2 (cit. on p. 28).

[RG19] N. Reimers and I. Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks." In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Nov. 2019. URL: https://arxiv.org/abs/1908.10084 (cit. on pp. 38, 41).

[SAS24] R. Stureborg, D. Alikaniotis, and Y. Suhara. *Large Language Models are Inconsistent and Biased Evaluators*. May 2, 2024. URL: http://arxiv.org/pdf/2405.01724v1 (cit. on p. 90).

[Sch+24] S. Schulhoff et al. *The Prompt Report: A Systematic Survey of Prompt Engineering Techniques*. June 6, 2024. URL: http://arxiv.org/pdf/2406.06608v6 (cit. on pp. 1, 12–16, 24–26).

[SDP20] T. Sellam, D. Das, and A. P. Parikh. *BLEURT: Learning Robust Metrics for Text Generation*. Apr. 9, 2020. URL: http://arxiv.org/pdf/2004.04696v5 (cit. on p. 28).

[Sho+24] O. Shorinwa et al. *A Survey on Uncertainty Quantification of Large Language Models: Taxonomy, Open Research Challenges, and Future Directions*. 2024. arXiv: 2412.05563 [cs.CL]. URL: https://arxiv.org/abs/2412.05563 (cit. on p. 13).

[SLL25] A. R. Sabau, D. Lammers, and H. Lichter. "SecuRe–An Approach to Recommending Security Design Patterns." In: *arXiv preprint arXiv:2501.14973* (2025) (cit. on pp. 6–8).

[Sun+25] F. Sun et al. *Large Language Models are overconfident and amplify human bias*. May 4, 2025. URL: http://arxiv.org/pdf/2505.02151v1 (cit. on p. 12).

[Tia+23] K. Tian et al. *Just Ask for Calibration: Strategies for Eliciting Calibrated Confidence Scores from Language Models Fine-Tuned with Human Feedback*. May 24, 2023. URL: http://arxiv.org/pdf/2305.14975v2 (cit. on p. 12).

[Ton+24] S. M. T. I. Tonmoy et al. *A Comprehensive Survey of Hallucination Mitigation Techniques in Large Language Models*. 2024. arXiv: 2401.01313 [cs.CL]. URL: https://arxiv.org/abs/2401.01313 (cit. on pp. 10, 11).

[Wei+22] J. Wei et al. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. Jan. 28, 2022. URL: http://arxiv.org/pdf/2201.11903v6 (cit. on pp. 10, 13, 15, 26).

[Wei+24] H. Wei et al. *Systematic Evaluation of LLM-as-a-Judge in LLM Alignment Tasks: Explainable Metrics and Diverse Prompt Templates*. Aug. 23, 2024. URL: http://arxiv.org/pdf/2408.13006v1 (cit. on pp. 16, 27).

[Whi+23] J. White et al. *A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT*. 2023. DOI: 10.48550/ARXIV.2302.11382 (cit. on pp. 1, 13, 15, 24, 25).

[Xio+23] M. Xiong et al. *Can LLMs Express Their Uncertainty? An Empirical Evaluation of Confidence Elicitation in LLMs*. June 22, 2023. URL: http://arxiv.org/pdf/2306.13063v2 (cit. on p. 12).

[Yao+23] S. Yao et al. *Tree of Thoughts: Deliberate Problem Solving with Large Language Models*. May 18, 2023. URL: http://arxiv.org/pdf/2305.10601v2 (cit. on p. 26).

[YRA24] J. S. Yeow, M. E. Rana, and N. A. Abdul Majid. "An Automated Model of Software Requirement Engineering Using GPT-3.5." In: *2024 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems (ICETSIS)*. 2024, pp. 1746–1755. DOI: 10.1109/ICETSIS61505.2024.10459458 (cit. on pp. 75, 78).

[Zha+19a] T. Zhang et al. *BERTScore: Evaluating Text Generation with BERT*. Apr. 22, 2019. URL: http://arxiv.org/pdf/1904.09675v3 (cit. on p. 28).

[Zha+19b] W. Zhao et al. *MoverScore: Text Generation Evaluating with Contextualized Embeddings and Earth Mover Distance*. Sept. 5, 2019. URL: http://arxiv.org/pdf/1909.02622v2 (cit. on p. 28).

[Zha+22] Z. Zhang et al. *Automatic Chain of Thought Prompting in Large Language Models*. Oct. 7, 2022. URL: http://arxiv.org/pdf/2210.03493v1 (cit. on pp. 15, 26).

[Zho+22] D. Zhou et al. *Least-to-Most Prompting Enables Complex Reasoning in Large Language Models*. May 21, 2022. URL: http://arxiv.org/pdf/2205.10625v3 (cit. on p. 26).

Glossary

IndDom Industry Domain

SSRS Synthetic System Requirement Specification

SSRS-Gen Synthetic System Requirement Specification Generator

SyRS System Requirement Specification