CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

LLM-Assisted Requirements Decomposition in Automotive Software Engineering

A Case Study at Volvo Cars

Master's Thesis in Computer science and engineering

Nhât Nam Hoàng, Melker Rååd

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2026

Master's Thesis 2026

LLM-Assisted Requirements Decomposition in Automotive Software Engineering

A Case Study at Volvo Cars

Nhât Nam Hoàng, Melker Rååd

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2026

LLM-Assisted Requirements Decomposition in Automotive Software Engineering
Nhât Nam Hoàng, Melker Rååd

Supervisor: Ali Nouri, Department of Computer Science and Engineering
Examiner: Philipp Leitner, Department of Computer Science and Engineering

Master's Thesis 2026
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2026

Nhât Nam Hoàng, Melker Rååd
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Requirements decomposition in automotive software engineering is a complex and context-dependent activity that requires architectural understanding, abstraction-level judgement, and domain expertise. Although Large Language Models (LLMs) have shown potential in several software engineering tasks, their use for requirements decomposition in safety-critical automotive contexts remains insufficiently understood.

This thesis investigates how LLM-based assistants can support requirements decomposition in automotive software engineering through a case study at Volvo Cars. The study follows a Design Science Research methodology combining semi-structured interviews, iterative artefact development, and industrial evaluation with domain experts. The resulting human-in-the-loop artefact combines contextual grounding through standards, historical decompositions, and system-structure information with hierarchy guidance, structured prompt orchestration, and schema-constrained generation.

The artefact was evaluated through benchmark-based assessments, contextual evaluations using participant-selected requirements, and qualitative feedback sessions with automotive domain experts. The results suggest that LLM-based assistants can support requirements decomposition by reducing cognitive effort, improving contextual awareness, and providing alternative decomposition perspectives during refinement. The evaluation further indicates that contextual grounding, hierarchy guidance, and structured orchestration positively influenced expert-perceived output quality and reviewability.

At the same time, important limitations remain related to abstraction-level consistency, incomplete contextual understanding, and the need for expert validation in safety-critical engineering contexts. The findings suggest that LLM-based decomposition assistants are most useful as decision-support tools within human-in-the-loop workflows rather than as autonomous requirement generation systems.

Keywords: Requirements Engineering, Requirements Decomposition, Large Language Models, Human-in-the-Loop, Retrieval-Augmented Generation, Knowledge Graphs, Multi-Agent Systems, Automotive Software Engineering

Acknowledgements

We would like to thank our academic supervisor Ali Nouri for his feedback and colourful metaphors that helped us steer our thesis direction.

A special thanks to our industrial supervisor Tayssir Bouraffa who went above and beyond her role to support us for the thesis. The incredible feedback and support we received was invaluable to us.

Furthermore, we would like to thank Cem Soyulmaz & Mitko Arsov for their extensive technical knowledge and availability throughout the thesis.

Last but not least, we would like to thank the people at Volvo Cars who either participated in our thesis or enabled us to understand the inner workings within Volvo Cars.

Nhât Nam Hoàng, Melker Rååd, Gothenburg, June 2026

Contents

List of Figures

List of Tables

Abbreviations

Abbreviation	Meaning
CF	Customer Function
DO	Design Objective
DSR	Design Science Research
genAI	Generative Artificial Intelligence
HITL	Human-in-the-Loop
KDP	Key Decision Point
KG	Knowledge Graph
KG-RAG	Knowledge Graph Retrieval-Augmented Generation
LLM	Large Language Model
MAS	Multi-Agent System
PC	Product Capability
PF	Product Function
PSE	Product Structure Entity
RAG	Retrieval-Augmented Generation
RE	Requirements Engineering
RMS	Requirements Management System
RQ	Research Question
SE	Software Engineering
TAM	Technology Acceptance Model
YOE	Years of Experience

1

Introduction

Requirements Engineering (RE) in automotive software development involves the systematic refinement of requirements across multiple abstraction levels. High-level requirements are often expressed from a user or vehicle perspective and progressively refined into subsystem- and component-level specifications to support development, verification, and validation [1]. This refinement process requires engineers to preserve the intent of higher-level requirements while introducing increasing technical detail at lower levels.

The complexity of this work is increased by the coexistence of multiple requirement representations. Structured natural language is commonly used to express system properties, use cases describe interactions and scenarios during early design, and model-based representations may be introduced later to support detailed system design. Although these representations address different engineering needs, their coexistence increases the cognitive and organisational complexity of RE activities. Engineers must reason across abstraction levels and representation types while maintaining consistency, traceability, and alignment with the original requirement intent [1].

In the industrial context studied in this thesis, automotive software development at Volvo Cars is supported by a structured specification data architecture that organises requirements, tests, and related artefacts into a multi-level hierarchy. This hierarchy consists of five primary abstraction levels: Customer Function (CF), Product Function (PF), Product Capability (PC), Group of Components, and Component. At the highest level, requirements describe vehicle behaviour from the end-user's perspective, while lower levels introduce increasing technical specificity and eventually define implementation-relevant constraints. The hierarchy is intended to support systematic requirements decomposition and traceability from customer needs to technical realisation through separation of concerns and ownership across levels.

However, the existence of a structured hierarchy does not make decomposition straightforward. Decomposing requirements across multiple abstraction levels requires domain knowledge, architectural understanding, coordination between roles, and careful interpretation of business, technical, and verification-related constraints. Requirements decomposition therefore remains a demanding expert-driven activity. This motivates the investigation of assistive tooling that can help engineers manage

contextual information, preserve requirement intent, and produce candidate decompositions while keeping human judgement central to the process.

1.1 Problem Description and Research Gap

In industrial automotive software development, requirements decomposition is still largely performed manually by system engineers. Engineers must interpret high-level customer- or vehicle-oriented requirements, determine the appropriate abstraction level for derived requirements, allocate responsibilities across system elements, and preserve traceability to higher-level intent, verification needs, and technical constraints. As system scale and software complexity increase, this work becomes time-consuming, cognitively demanding, and vulnerable to inconsistencies, omissions, and weakened traceability [1, 2, 3].

Large Language Models (LLM) offer new opportunities for supporting natural-language intensive software engineering (SE) activities and have been increasingly adopted in SE tasks, especially in later lifecycle phases [4]. However, their use in early-phase requirements engineering, and particularly in requirements decomposition for safety-critical automotive systems, remains insufficiently understood. This is problematic because requirements decomposition requires more than fluent requirement text: generated outputs must preserve intent, fit the target abstraction level, reflect domain context, and remain subject to expert validation.

This study addresses the need to understand how LLM-based assistants can be designed and evaluated as support tools for automotive requirements decomposition. The focus is on assistive generation of candidate decompositions and contextual support, rather than autonomous requirement approval or formal verification.

1.2 Purpose of the Study

The purpose of this study is to investigate how an LLM-based assistant can support requirements decomposition in automotive SE. The study focuses on the design, refinement, and evaluation of an artefact that assists engineers in transforming higher-level requirements into structured lower-level candidate requirements within an industrial requirements hierarchy. The scope of the study is limited to assistive support for decomposition rather than autonomous requirement approval or formal verification. The artefact is intended to provide candidate outputs, contextual information, and explanatory support that engineers can inspect, revise, or reject. The evaluation therefore focuses on expert-perceived usefulness, output quality, workflow fit, and limitations in an industrial case-study setting, rather than proving the formal correctness or completeness of generated requirements.

1.3 Research Approach, Key Findings, and Contributions

This study follows a Design Science Research (DSR) approach grounded in the methodologies proposed by Hevner & Chatterjee [5] and Wieringa [6]. The problem investigation was conducted through semi-structured interviews with automotive domain experts concerning current decomposition practices, perceived opportunities for LLM support, and risks associated with generative AI-assisted decomposition. The interview findings informed a set of design objectives that guided the iterative development of a decomposition assistant artefact.

The resulting artefact combines contextual grounding through standards, historical decompositions, and system-structure information with hierarchy guidance, structured prompt orchestration, and schema-constrained output generation. The artefact was evaluated by domain experts through a benchmark assessment, a contextual assessment using participant-selected requirements, and open-ended feedback on usefulness, risks, and adoption conditions.

The findings suggest that LLM-based assistants can support requirements decomposition by reducing cognitive effort, providing alternative decomposition perspectives, supporting exploratory refinement, and improving the clarity and consistency of generated requirements. At the same time, important limitations remained related to abstraction-level uncertainty, incomplete contextual understanding, risks of overlooking safety- or verification-relevant details, and the continued need for expert validation. The artefact was generally perceived as useful for supporting expert-led decomposition workflows. Contextual grounding mechanisms based on standards and system context were assessed more positively, while historical decomposition reuse showed greater dependence on contextual interpretation and applicability assessment. Overall, the results suggest that LLM-based decomposition assistants are most appropriate as human-in-the-loop (HITL) decision-support tools rather than autonomous requirements generators.

The contribution of this thesis is threefold. First, it provides an empirical characterisation of requirements decomposition challenges and practitioner expectations regarding LLM support in an automotive systems engineering context. Second, it presents a set of empirically grounded design objectives together with an artefact design for LLM-assisted requirements decomposition. Third, it reports an exploratory industrial evaluation of the artefact and identifies perceived benefits, limitations, and adoption considerations for future automotive software requirements engineering workflows.

1.4 Research Questions

Based on the identified research gap and the objectives of the study, the following research questions (RQ) were formulated to guide the design and evaluation of the

proposed artefact. The first research question focuses on the perceived benefits and limitations of LLM-assisted requirements decomposition, while the second examines how different artefact design choices influence expert-perceived decomposition quality.

RQ1: What are the benefits and limitations of using LLM-based assistants to support requirements decomposition in automotive software development?

RQ2: How do design choices in an LLM-based assistant influence expert-perceived quality in requirements decomposition outputs?

2

Background

This chapter introduces the domain and technical concepts needed to understand the artefact design presented in this thesis. The chapter first presents RE as the broader discipline in which requirements are elicited, specified, validated, and traced. It then narrows the focus to requirements decomposition in automotive SE, where requirements must be refined across abstraction levels while preserving intent, traceability, and alignment with system architecture. The second part of the chapter introduces LLMs and the mechanisms used to make their outputs more suitable for the requirements decomposition process.

2.1 Requirements Engineering in Software Engineering

RE comprises the systematic activities and artefacts used to establish what a system must achieve and the constraints under which it must operate throughout its life cycle. RE is commonly described as a set of recurring and interwoven activities, including stakeholder identification, elicitation, specification, analysis, validation, and management of requirements [7, 8, 9]. These activities transform stakeholder needs into agreed and documented requirements that can guide design, implementation, verification, and later system evolution [7, 8, 10].

RE is critical in SE because requirements act as the primary reference for architectural decisions, implementation planning, and verification activities [9]. Additionally, defects in requirements (e.g., ambiguity, inconsistency, omissions) propagate downstream and often lead to rework and expedient implementation compromises. The resulting quality issues can accumulate as technical debt. Since such issues are typically more expensive to correct later in the life cycle, RE must support early clarification, validation, and controlled evolution of requirements as stakeholder needs, constraints, and operational contexts change [7, 10].

A central concern in RE is therefore the quality of requirements. High-quality requirements are expected to be understandable, consistent, complete, feasible, verifiable, and traceable [10]. These qualities are especially important when requirements are refined or decomposed, because ambiguities or omissions at higher levels can be amplified when lower-level requirements are derived [9, 10]. For LLM-assisted

decomposition, these qualities are particularly important because generated requirements should support expert review and further refinement rather than be accepted as final specifications [11, 12, 13].

In large software-intensive systems, requirements are also interdependent and must be related to their sources and downstream artefacts. Requirements traceability supports accountability and change impact analysis by maintaining links between related requirements, their purpose, and verification artefacts [14]. These RE foundations become increasingly important in domains where requirements are refined across multiple abstraction levels and where development is constrained by safety, regulation, and organisational complexity [2].

2.2 Requirements Engineering in the Automotive Industry

Requirements engineering in the automotive industry is shaped by safety, cybersecurity, legal, and organisational constraints. Automotive systems are software-intensive and often safety-critical, which means that requirements must support not only implementation but also verification, validation, compliance, and change impact analysis. Legal and safety constraints influence the RE workflow and can slow down development processes [15]. Standards such as ISO 26262 [16], for functional safety, and ISO/SAE 21434 [17], for cybersecurity engineering, place strong demands on how requirements are specified, traced, reviewed, and maintained.

A key characteristic of automotive RE is that requirements are refined across multiple abstraction levels and organisational boundaries. High-level vehicle or customer-oriented requirements must be translated into more technical requirements for functions, systems, modules, and components. This refinement requires engineers to interpret the original intent and allocate responsibilities to the appropriate system elements. Furthermore, they must ensure that derived requirements remain aligned with safety, cybersecurity, architectural, and verification constraints. In addition, automotive RE is affected by communication and coordination challenges [2].

Traceability is therefore central in automotive RE, supporting both forward and backward traceability relationships [18]. It supports compliance, impact analysis, accountability, and validation activities [18, 19]. These characteristics make automotive requirements decomposition a context-dependent activity that requires both textual requirement understanding and awareness of relationships between requirements, system elements, and constraints.

These characteristics are also reflected in the information structures used within industrial automotive development processes. At Volvo Cars, Product Structure Entities (PSEs) are used to represent and organise product-related structures, responsibilities, and engineering artefacts across different hierarchy levels. Each PSE is associated with a specific abstraction level and may contain linked requirements, metadata, and related engineering information. PSEs support the allocation and

traceability of requirements across subsystem boundaries and provide contextual information about architectural scope, system responsibilities, and intended decomposition levels. This makes PSEs relevant as a source of contextual grounding for requirements decomposition activities.

2.3 Requirements Decomposition

Requirements decomposition concerns the practice of deriving lower-level requirements from higher-level requirements [20]. This supports the gradual transition from broad system or stakeholder intentions towards requirements that can guide design, implementation, verification, and validation activities [1, 21]. The derived requirements typically introduce additional technical specificity and may be associated with particular functions, subsystems, or components, depending on the structure of the engineering organisation and system architecture [1].

In this thesis, requirements decomposition is defined as the process of transforming a requirement into a set of lower-level requirements across abstraction levels, while preserving the intent of the decomposed requirement and aligning the derived requirements with system architecture, traceability needs, and the intended target level. This definition builds on the view that decomposition is not merely the act of splitting one requirement into several smaller statements. Rather, it is an engineering activity that involves interpretation of purpose, allocation of responsibilities, and consideration of system constraints [21].

A central challenge in requirements decomposition is managing the transition between abstraction levels. Higher-level requirements (parents) are often written from a stakeholder, user, or vehicle perspective, while lower-level requirements (children) introduce increasing technical specificity [1, 21]. The semantic shift from stakeholder-oriented intent to technical realisation changes both the content and the expected level of detail of the requirement [21]. If a derived requirement remains too abstract, it may provide insufficient guidance for downstream engineering activities; if it becomes too detailed too early, it may prescribe design decisions at an inappropriate level of the requirements hierarchy [1, 21].

Traceability is also affected during decomposition. Maintaining such relationships can be challenging because requirement structures do not always form a strict tree. A higher-level requirement may be refined into several lower-level requirements, while a derived requirement or logical element may contribute to several PSEs [18, 20]. Figure 2.1 illustrates this requirement growth and many-to-many traceability. Such relationships are important for impact analysis, verification planning, consistency management, and compliance in safety-critical domains [18, 19]. These characteristics make requirements decomposition dependent on domain expertise, architectural context, and coordination across engineering roles [1, 2]. A useful decomposition must preserve parent intent, introduce an appropriate level of technical detail, allocate responsibilities coherently, and maintain traceability across levels [18, 20, 21]. Within Volvo Cars, the process is commonly referred to as requirements breakdown,

although the term requirements decomposition is used throughout this thesis to align with the academic literature.

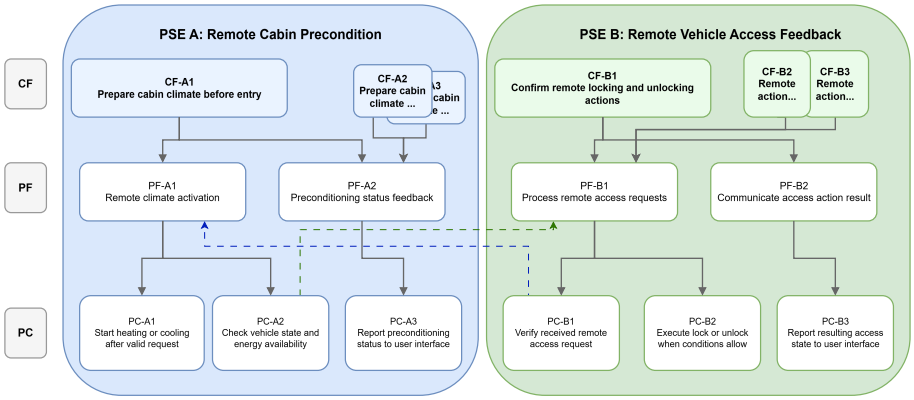

Figure 2.1: Synthetic illustration of requirements decomposition across CF, PF, and PC levels. Two fictional PSEs are shown to demonstrate how high-level customer-facing intent can branch into lower-level functional responsibilities and more specific product capabilities. Solid arrows indicate direct decomposition links, while dashed arrows illustrate possible cross-PSE dependencies or shared contextual relationships. The example is illustrative only and does not represent an industrial requirement or Volvo Cars implementation.

2.4 Large Language Models

LLMs are neural language models trained on large text corpora to process and generate natural language. Many modern LLMs are based on the Transformer architecture, which uses self-attention to model relationships between tokens in an input sequence [22, 23]. Compared with earlier sequential architectures, Transformer-based models are better suited to capturing dependencies across longer text sequences, which has contributed to their use in a wide range of natural language processing tasks [22, 23]. The capabilities of modern LLMs are associated with large-scale pre-training on extensive text corpora and, in some cases, models with billions of trainable parameters [24, 25]. Through prompting, task-specific behaviour can often be elicited without changing the underlying model parameters [12, 13]. Since requirements are commonly represented in natural language, LLMs are relevant for RE activities that involve interpreting, reformulating, summarising, and structuring textual information [7, 8, 9].

LLMs also have well-documented limitations. One common limitation is hallucination, where a model produces output that is syntactically plausible but factually or logically incorrect [11]. This limitation is related to the probabilistic nature of LLM generation and the fact that models generate text based on learned patterns rather than direct verification against external reality [11, 12, 26]. In safety-critical engineering contexts, such unsupported or incorrect outputs can be problematic if they are not reviewed by domain experts [11].

A second limitation is that the knowledge available to an LLM through its parameters is limited by the data and time of pre-training. As a result, LLMs may lack access to domain-specific documentation, recent project decisions, internal requirements, or organisation-specific engineering conventions [27]. This is particularly relevant for automotive requirements decomposition, where useful output may depend on company-specific hierarchies, standards, released requirements, and architectural relationships.

A third limitation concerns the context window, which defines the amount of text that a model can process in a single interaction [22]. Since RE tasks may involve standards, guidelines, historical requirements, architectural information, and user input, relevant information may exceed what can be provided directly in a single prompt. In such cases, important constraints may be omitted, truncated, or not effectively used by the model.

These limitations motivate the use of additional mechanisms for controlling and grounding LLM-assisted requirements decomposition. Prompt engineering can structure instructions and outputs, retrieval-augmented generation can provide relevant textual context, knowledge graphs can preserve relationships between requirements and related artefacts, and multi-agent orchestration can separate the workflow into inspectable stages. The following sections introduce these mechanisms.

2.5 Prompt Engineering

Prompt engineering refers to the deliberate design of prompts used to guide an LLM towards a desired task, response style, or output structure [12, 28]. Prompts may contain task instructions, contextual information, constraints, examples, and output format specifications. Since LLM behaviour is sensitive to how tasks and context are specified, prompt design can reduce ambiguity and improve alignment between generated outputs and the intended task [28, 29].

Prompt engineering is particularly relevant for requirements decomposition because multiple contextual sources must often be handled simultaneously, including the source requirement, decomposition objectives, hierarchy-related guidance, retrieved standards, historical decompositions, and expected output structures. Structured separation of prompt components can help distinguish instructions, contextual information, constraints, and output requirements, while direct task instructions and negative constraints may reduce unsupported or irrelevant outputs [12, 13]. Output generation can additionally be constrained to predefined structures containing candidate requirements and supporting explanatory elements, improving reviewability during RE activities.

However, prompt engineering alone cannot compensate for missing or insufficient context. If relevant standards, architectural information, historical requirements, or system-specific constraints are unavailable, even well-structured prompts may produce incomplete or weakly grounded outputs. These limitations motivate the

use of complementary contextual grounding and orchestration mechanisms.

Additional prompt engineering principles and implementation-oriented techniques considered in the artefact design are presented in Appendix A.

2.6 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) [30] is a framework for improving the output of generative AI models by supplying them with retrieved, contextually relevant information. A RAG system consists of two primary components: a retriever and a generator. When a query is provided, the retriever performs a similarity search over embedded text chunks in a vector database and returns the most relevant information. This retrieved context is then provided to the generative model, improving alignment with existing documentation and reducing hallucination [31].

The retrieval component typically relies on semantic similarity using text embeddings, where natural language is represented as dense vectors in a high-dimensional space. Semantically similar texts are located closer together in this space and can be compared using similarity metrics such as cosine similarity [32]. This enables retrieval of semantically similar text even when they are phrased differently, which is particularly useful in requirements engineering due to the variability and ambiguity of natural language [33]. Retrieval can also be augmented, for example through context-enriched chunking, where document-level information is included in each chunk before embedding [34]. This can guide retrieval towards appropriate source documents rather than relying solely on local semantic similarity.

In practice, RAG systems have been shown to be effective for retrieving information from large collections of technical documentation [35]. However, they also exhibit important limitations. Relevant information may not be retrieved if queries are ambiguous, embeddings fail to capture the intended semantic meaning, or retrieved chunks lack sufficient contextual information. Even when relevant information is retrieved, the generative model may fail to correctly interpret, prioritise, or apply the provided context during generation [36]. As a result, RAG-based systems may still produce incomplete, weakly grounded, or inconsistent outputs despite access to relevant source material.

2.7 Knowledge Graphs

A knowledge graph represents real-world knowledge as a network of entities and their relationships, where nodes correspond to entities and edges represent relationships between them [37]. Entities can represent objects, systems, or abstract concepts, while relationships capture how these entities are connected.

Knowledge graphs typically represent data as triples of the form (s, p, o), where s is the subject, p is the predicate, and o is the object. For example, (Gothenburg, city, Sweden) expresses a relationship between two entities [37]. In practice, knowledge

graphs are often implemented using graph databases, which enable efficient storage and querying of graph-structured data [38].

A key advantage of graph-based representations is their ability to efficiently traverse relationships between entities. Unlike relational systems that rely on JOIN operations, graph databases support index-free adjacency, where each node directly references its connected neighbours. This enables efficient exploration of multi-level relationships, making knowledge graphs particularly suitable for representing dependencies and hierarchical structures in requirements engineering [37]. This structured representation complements retrieval-based approaches by providing explicit system-level relationships that are not captured through similarity-based methods.

2.8 Multi-Agent Systems and Orchestration

LLMs have demonstrated significant advances across multiple natural language processing disciplines, but their performance degrades in more complex multi-step or collaborative problems [39]. To address these challenges, multi-agent-systems (MAS) are used to take actions and look for solutions similar to human collaboration [39, 40]. A MAS consists of multiple agents, where each agent takes on a specialised role with specific skills and reasoning patterns that collaborate with each other which leads to an increased intelligence compared to stand-alone LLMs [39, 41]. Furthermore, an agent can be bound by a contract that defines how it may perform a task, including its input/output expectations, available skills, resource limits, time limits, success criteria, and termination conditions. In this sense, contracts make agent behaviour more predictable and auditable by constraining not only what an agent should do, but also how much it may consume and when execution must stop [42].

In the context of MAS, orchestration refers to the automated coordination of specialised LLM agents to treat a prompt as a multi-layered task rather than a single monolithic output generation. An orchestrator is responsible to call different agents depending on circumstances during generation and that each agent works in collaboration instead of isolation [39]. This framework enhances computational efficiency and reasoning depth by reducing complexity of individual task execution [43].

3

Related Work

LLM-assisted RE has gained increasing attention as a way to support language-intensive engineering activities, yet its use for requirements decomposition in industrial automotive software settings remains insufficiently established. Prior work relevant to this thesis can be grouped into four areas: decomposition challenges in safety-critical and automotive engineering, LLM applications in RE, LLM-assisted requirement generation and decomposition, as well as mechanisms for grounding and controlling LLM-based engineering tools.

3.1 Decomposition as a Requirements Engineering Challenge

Requirements decomposition remains a persistent challenge in automotive engineering. Prior work identifies weak traceability, redundancy, inconsistent refinement across abstraction levels, and difficulty managing system complexity as recurring issues in manual decomposition processes [2, 3]. These problems are particularly relevant in automotive SE, where requirements are refined across several abstraction levels and must remain aligned with system architecture, verification needs, and organisational responsibilities.

The literature also indicates that decomposition problems are not only caused by the wording of individual requirements. They emerge from the interaction between technical refinement, coordination between roles, and process constraints. Inconsistent changes at lower abstraction levels can introduce gaps or contradictions relative to higher-level intent, while formal RE processes may be perceived as too slow for day-to-day engineering work, encouraging informal or ad-hoc requirement practices [2, 44]. This suggests that decomposition support must address more than textual reformulation. It must also support context awareness, traceability, and reviewability.

Recent work on decomposition-support tools therefore points towards semi-automated support rather than full automation. Semi-automated approaches have been proposed to improve the quality and verifiability of lower-level requirements, while empirical studies show that practitioners continue to face challenges when refining abstract safety concerns into more concrete artefacts while preserving traceability

[45, 46]. Existing research therefore frames decomposition as an expert-led activity that may benefit from automation, provided that engineers remain responsible for interpreting, validating, and refining the resulting requirements.

3.2 Large Language Models as Assistive Tools in Requirements Engineering

Research on LLMs in SE has so far focused more strongly on implementation and maintenance than on early RE activities [4]. Practitioner use follows a similar pattern, with genAI tools being used more frequently for coding and debugging than for planning or requirements-related work [47]. RE is therefore a comparatively less mature application area for LLMs, despite the fact that many RE tasks involve the interpretation, transformation, and validation of natural-language artefacts.

Within RE, existing studies suggest that LLMs can support several language-intensive tasks. Prior work has explored their use for eliciting requirements from user reviews and stakeholder interviews, generating test cases from requirements, and supporting traceability-related analysis. Across these tasks, the main role of LLMs is not to replace engineering judgement, but to assist with interpreting, reformulating, linking, or generating candidate artefacts from textual input [48, 49, 50, 51].

A similar pattern appears in work on downstream artefact generation from requirements. LLMs can transform requirement-like inputs into code, tests, or executable specifications, but the resulting outputs require guardrails, validation, or expert curation before they can be used in engineering workflows [52, 53].

3.3 Large Language Models-Assisted Requirement Generation and Decomposition

Research closer to requirement generation and decomposition shows that LLMs can produce useful candidate requirement content when provided with suitable context. In compliance-oriented settings, LLMs have been used to derive more actionable requirement content from standards, controls, and dense technical documentation [13, 54, 55, 56]. These studies indicate that LLMs can help surface relevant requirement areas from complex information sources, while still relying on human review to assess correctness, relevance, and completeness.

Existing studies in adjacent high-assurance domains suggest that LLMs can support the derivation of lower-level requirements or decomposition-adjacent artefacts, but they also show that performance depends on the decomposition level, available context, and validation strategy . In particular, reported issues around abstraction-level variation, traceability, and expert validation suggest that generated requirements are best treated as candidate outputs rather than finalised specifications [55, 57, 58].

For this thesis, the most relevant implication is that existing work has not yet estab-

lished how LLM-generated candidate decompositions can be aligned with industrial automotive requirement hierarchies. Prior studies provide evidence that LLMs can support requirement generation and decomposition-related activities, but they leave open how such support should incorporate automotive-specific abstraction levels, company-specific product context, historical decomposition knowledge, and expert review practices.

3.4 Grounding, Control, and Human Validation in Large Language Models-Based Requirements Engineering

A recurring response to the limitations of LLMs is to combine generation with mechanisms for grounding, control, and validation. Retrieval-based approaches are used to connect model outputs to external documentation, while graph-based approaches add relational structure that can support tasks involving dependencies, hierarchies, and cross-document connections [30, 59]. In safety-critical industrial contexts, KG-enhanced retrieval has shown value for tasks that require structured reasoning over interconnected engineering knowledge, such as failure analysis [60]. Other KG-RAG approaches similarly aim to improve domain-specific reasoning by combining retrieved evidence with explicit entity relationships [61, 62]. Notably, these works create KGs during runtime, unlike our artefact.

Other work focuses on controlling the generation process itself. MAS and orchestration have been used to divide complex tasks into specialised subtasks, introduce intermediate checks, and simulate role-based reasoning [38, 43]. In RE, agent-based approaches have been applied to user-story prioritisation and debate-like requirements analysis, reflecting the fact that RE often involves multiple perspectives rather than a single deterministic transformation [63, 64]. HITL multi-agent approaches further emphasise that domain expertise remains necessary when applying LLMs to RE tasks [65].

Prompt structure also affects the quality and reviewability of generated requirements. Prior work shows that explicit roles, structured inputs, rationales, and controlled output formats can improve generated requirement quality, while automatic prompt optimisation may improve average performance without eliminating the need for human judgement [13, 66]. Across these approaches, the literature points towards the same general design principle: LLM-based RE tools are more useful when their outputs are grounded in relevant context, constrained by task-specific structure, and made inspectable for expert review.

3.5 Research Gap

The reviewed literature shows that requirements decomposition remains difficult in automotive and safety-critical engineering because it requires abstraction-level

judgement, traceability, domain knowledge, and coordination across engineering roles [2, 3, 46]. It also shows that LLMs can support several RE activities, including elicitation, traceability analysis, compliance-oriented generation, test generation, and decomposition-related tasks [13, 48, 49, 50, 51, 55, 56]. In addition, prior work demonstrates that grounding and control mechanisms such as RAG, KG-enhanced retrieval, multi-agent orchestration, structured prompting, and HITL validation can improve the contextual relevance and reviewability of LLM outputs [30, 59, 65, 66].

However, existing research provides limited evidence on how these mechanisms can be combined and evaluated as an HITL assistant for requirements decomposition within an industrial automotive requirements hierarchy. In particular, prior work does not sufficiently address how such an assistant should incorporate company-specific product structures, standards-oriented retrieval, historical decomposition reuse, hierarchy guidance, structured prompt orchestration, and expert validation in one workflow. To the best of our knowledge, the specific combination of semantic retrieval, graph-based decomposition reconstruction, and human-validated historical decomposition reuse proposed in this thesis has not previously been evaluated for automotive requirements decomposition.

This thesis addresses this gap through a case study at Volvo Cars by designing and evaluating an LLM-based artefact for automotive requirements decomposition.The artefact combines standards-oriented retrieval, historical decomposition reuse through KG-RAG, PSE-based contextual summarisation, hierarchy guidance, structured prompt orchestration, and schema-constrained output generation. The contribution lies in investigating how these mechanisms can support expert-led generation and review of candidate lower-level requirements in an industrial automotive RE context.

4

Methodology

This research adopts a DSR methodology to address the gap between capabilities of LLMs and requirements decomposition in the automotive software engineering domain. DSR is particularly suited for this study, as the objective is not only to understand a problem but to design and evaluate an artefact that provides a practical solution within an industrial context.

4.1 Overview

The research design is grounded in the Three-Cycle Model proposed by Hevner & Chatterjee [5] and is further operationalised through the engineering cycle described by Wieringa [6]. Together, these frameworks provide complementary perspectives: Hevner & Chatterjee's model ensures alignment between the research, its industrial environment, and the existing knowledge base, while Wieringa's cycle structures the process of problem investigation, solution design, implementation, and evaluation. A general overview can be seen in Figure 4.1.

Following Hevner & Chatterjee's three cycle model, this study is structured around three interrelated cycles:

- **Relevance Cycle:** Connects the research to the industrial environment. In this study, the relevance cycle is operationalised through semi-structured interviews with domain experts at Volvo Cars, which serve to identify the challenges associated with requirements decomposition and to establish the DOs for the artefact.

- **Design Cycle:** Represents the iterative process of building and refining the artefact. The core of this research lies in this cycle, where a multi-agent system for requirements decomposition is developed through successive iterations, progressing from an initial integration prototype to a RAG approach, followed by a Knowledge Graph-enhanced RAG architecture, and ultimately a fully optimised multi-agent system.

- **Rigour Cycle:** Anchors the research in existing scientific knowledge. The design of the artefact is informed by prior work in LLMs, retrieval-augmented generation, knowledge graphs, and multi-agent systems. These knowledge

sources provide the theoretical grounding for design decisions and enable the interpretation of results.

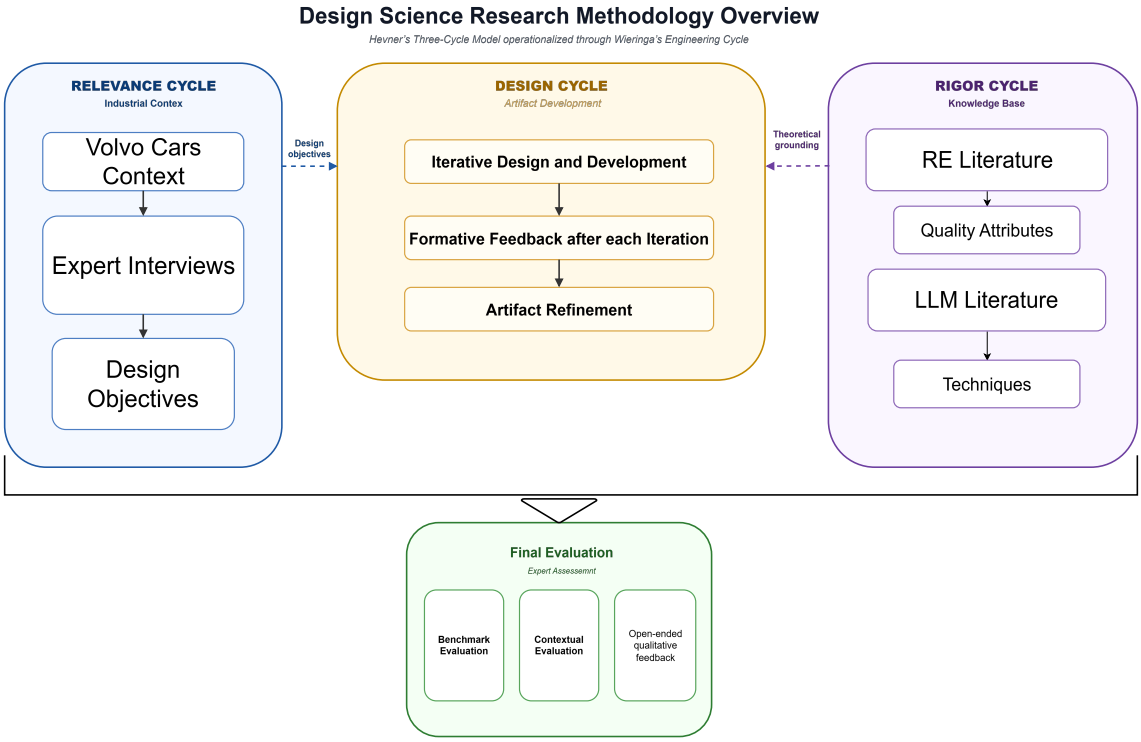

Figure 4.1: Overview of the DSR Methodology.

To operationalise the **design cycle**, this study follows Wieringa's engineering cycle, which structures the research into four phases:

1. Problem Investigation: Understanding the industrial problem through empirical data collection (expert interviews).

2. Treatment Design: Translating identified needs into DOs and architectural decisions.

3. Treatment Implementation: Developing the artefact through iterative design and refinement.

4. Treatment Evaluation: Assessing the utility of the artefact in its intended context.

A key characteristic of this research is the iterative problem–solution co-evolution during the design cycle. Rather than developing the artefact in a single pass, the system evolved through multiple iterations, where each version was subjected to formative evaluation to identify limitations and inform subsequent design decisions. This approach ensures that the final artefact is both grounded in the problem context and refined through continuous learning.

Different evaluation strategies were used. During the evaluation cycle, formative

evaluations were used to guide artefact development. At the end of the study, a summative evaluation was conducted to assess the artefact overall's utility, see Chapter 7. This enables systematic evaluation of both perceived usefulness and output quality across different conditions.

4.2 Participants

The empirical parts of this study involved domain experts from Volvo Cars who contributed either to the problem investigation, the final artefact evaluation, or both. Participants were selected based on their experience with requirements engineering, system design, SE, or related automotive development activities. The study used two main empirical data sources. Table 4.1 provides an overview of participant background and involvement across the two empirical phases. In total, fifteen unique participants were involved for the thesis. Thirteen of these participants took part in the relevance cycle (pilot interview was removed from the final results), eight of those returned for the evaluation.

Table 4.1: Participant Background: Years of Experience (YOE) in automotive industry, requirements engineering, and generative AI, including current roles. Experience values are approximate and based on participant self-reporting. Participation in the empirical phases is shown in column Interview and Evaluation. (P) denotes the pilot interview.

ID	Automotive	RE	genAI	Role	Interview	Evaluation
P1	2	2	1	Function Designer	Yes	Yes
P2	21	21	0	Senior Technical Lead	Yes	
P3	16	16	3	Function and Safety Designer	Yes	Yes
P4	4	4	1	System Designer	Yes	
P5	10	8	0	Function Designer	Yes	Yes
P6	7	5	3	Software Engineer	Yes	Yes
P7	25	25	0	System Designer	Yes	
P8	14	6	1	System Designer	Yes	
P9	6	6	1	Lead System Engineer	Yes	Yes
P10	8	5	1	System Architect	Yes	Yes
P11	7	14	1	Senior Technical Lead	Yes	
P12	9	2	1	System Designer	Yes	Yes
P13	11	11	4	Technical Expert	(P)	Yes
P14	15	15	1	System Architect		Yes
P15	9	5	1	System Designer		Yes
					($n_I = 13$)	($n_E = 10$)

4.3 Cycle 1: Problem Investigation (Relevance Cycle)

Following Wieringa's [6] engineering cycle, the research began with a systematic investigation of the design problem. The objective was to establish the relevance of the proposed artefact within the industrial context of Volvo Cars and to ensure that the research addresses a practically significant problem. More details can be read in Chapter 5.

Data Collection: Semi-Structured Interviews

To elicit domain-specific requirements, semi-structured interviews were conducted with 12 industry experts, including Systems Engineers and Requirements Leads. While the general problem area was identified beforehand through research, these interviews constituted the formal problem investigation phase.

Synthesis into Design Objectives

The interview transcripts were analysed using template analysis with a hybrid deductive-inductive coding strategy, as described in Section 5.5. The resulting themes captured process pain points, perceived benefits and limitations of LLM support, and artefact design expectations. These themes were then mapped to DOs in Section 5.8, which define the requirements and constraints for the artefact.

This step represents the transition from problem investigation to treatment design in Wieringa's engineering cycle and ensures traceability between the identified problem and the proposed solution.

The DOs served as the primary input to the subsequent design cycle, guiding architectural decisions and implementation priorities.

4.4 Cycle 2: Iterative Artefact Design (Design Cycle)

The artefact was developed through an iterative design process aligned with the Design Cycle in Hevner & Chatterjee's [5] Three-Cycle Model. Rather than a linear development process, the research followed an incremental problem–solution co-evolution approach, where each iteration of the artefact was refined based on observed limitations. More details about the artefact design and its respective cycles can be read in Chapter 6.

The artefact was developed within an existing multi-agent LLM assistant environment available within a requirement management system (RMS) in the industrial setting. This environment provided the general interaction and orchestration infrastructure for executing assistant workflows. However, the research artefact in

this study was not the surrounding assistant environment itself, but the dedicated requirements decomposition workflow designed, implemented, and evaluated within it. This distinction was important for scoping the design cycle: the existing environment served as the technical host for the artefact, while the iterative design work focused on decomposition-specific mechanisms such as contextual grounding, historical knowledge reuse, hierarchy support, and HITL validation.

Each iteration consisted of:

1. Implementation of a design increment
2. Formative evaluation of the artefact's performance
3. Identification of limitations
4. Refinement of the design in the subsequent iteration

This approach ensured continuous alignment between the artefact and the DOs derived in Cycle 1.

Formative Evaluation Strategy

During the design cycle, formative evaluations were conducted after each iteration to assess the artefact's ability to address identified requirements and to guide subsequent design decisions.

The evaluations were qualitative and exploratory in nature. They were focused on identifying limitations in reasoning, grounding, and output consistency. The form of assessment varied depending on the type of implementation being investigated. Technical behaviour, such as retrieval mechanism, prompt structure or workflow integration, was primarily conducted through author inspection of the artefact output. For more domain specific concerns, e.g., abstraction level correctness, automotive expert(s) were consulted. The expert could review the artefact behaviour and commented on the specific concern.

The purpose of these evaluations was not to produce generalisable performance metrics, but to support iterative refinement of the artefact. In line with the research objective, this study does not perform component-level or ablation-based evaluations. Instead, individual components were assessed only in terms of their contribution to the overall artefact behaviour. This decision reflects the DSR focus on evaluating the utility of the complete artefact rather than isolating and benchmarking individual techniques.

4.5 Cycle 3: Rigour Cycle

The rigour cycle anchors the research in the existing body of scientific knowledge and ensures that the design and evaluation of the artefact are grounded in established

theories and methods. In this study, the rigour cycle informed both the architectural design of the artefact and the interpretation of the evaluation results. The interaction between the rigour cycle and design cycle enabled a continuous refinement process, where theoretical insights were used to address empirically observed limitations. The following primary knowledge domains were considered:

- **Requirements Decomposition:** Foundational concepts from requirements engineering, particularly in requirements decomposition, traceability, and handling of complex system specifications, informed the problem framing and the definition of DOs. This domain knowledge ensured that the artefact aligns with established engineering practices and addresses real-world challenges in safety-critical automotive systems.

- **LLMs:** Prior research on LLM capabilities and limitations, particularly regarding reasoning, hallucination, and context sensitivity, informed the need for augmenting generative models with external knowledge sources.

- **RAG:** Existing work on retrieval-based augmentation guided the design of mechanisms for incorporating domain-specific documentation into the reasoning process, addressing limitations in factual grounding.

- **KGs:** Research on structured knowledge representation and relational reasoning informed the integration of a Knowledge Graph to model dependencies between requirements, components, and constraints.

- **MAS:** Concepts from multi-agent system design, including role specialisation and orchestration, influenced the decomposition of the system into multiple interacting agents to improve reasoning robustness and modularity.

Literature Selection

Two main approaches were used to find relevant literature for the rigour cycle. One was to query established academic databases such as arXiv, IEEE Xplore, Google Scholar, and ACM Digital Library. The following keyword combination were used to query the respective databases:
('Requirements Engineering' OR 'Requirements Decomposition') AND ('Large Language Models' OR 'Generative AI' OR 'LLM') AND ('Automotive' OR 'Safety-Critical'). Additionally, specific technical terms such as *'Retrieval-Augmented Generation'*, *(Knowledge Graphs)* and *(Multi-Agent-System)* were included to identify papers on architectural patterns for LLM integration.
Furthermore, the catalogue of the Requirements Engineering: Foundation for Software Quality (REFSQ) conference was scanned. For this, the two most recent paper catalogues from 2024 and 2025 were chosen. REFSQ was chosen as it is a prominent European conference regarding requirements engineering, specifically for the SE domain.

The relevant literature was initially scanned by looking at titles and abstracts to

discover relevance, selected articles were then read in detail to assess their methodology and to confirm relevance. Furthermore, the number of citations and published journals were also considered during the selection as significant number indicates credibility. Finally, snowball-search was applied by looking at the referenced papers in our selection which was inspired by systematic literature reviews guidelines by Wohlin [67].

5

Interviews

This chapter covers the process of the relevance cycle, presented in Section 4.3, where the core challenges of the requirements decomposition process were identified. To answer RQ1 and and base our decision making for RQ2, semi structured interviews were conducted to investigate and identify the problem domain. The interviews aimed to understand the current manual requirements decomposition process by identifying challenges and limitations, understand attributes of a quality requirement, and explore any prior experience with genAI tools which can be related to the process. The design followed the established guidelines on how to perform qualitative research by Runeson & Höst [68]. This chapter starts with the interview procedure, followed by the analysis of the resulting data. Finally, the chapter concludes with the DOs, we identified from the interviews.

5.1 Interview Method

After booking a time slot, the participant received an information sheet that contained the general information about the interview and informed the participant how their data is processed. In addition, a consent form was sent out that was later signed at the start of the interview. The interviews were conducted on Microsoft Teams and lasted about 60 minutes. Microsoft Teams was chosen to enable remote participation and the ability to produce an initial transcription draft. These two factors helped with flexibility and reducing transcription time. The transcription was then validated against the recording and verified by the corresponding participant. Before the interview began, an introductory video was played that reiterated the most important information required for the interview. After that, the participant's consent was collected for both the recording and interview process [69].
If the participant used terminology or acronyms that were unfamiliar to the authors, they were asked to elaborate.

5.2 Interview Guide

A detailed interview guide was created to guide the interviewer and to ensure consistency between participants and the collected data, but also to allow for exploration of unanticipated topics. Before the first interview, a pilot interview study was con-

ducted to finalise the guide. The pilot included all the steps leading up to the interview such as using the booking system, the interview itself, and finally the steps after the interview such as the transcription. The guide consisted of 14 main questions each with potential follow-up probes. The questions were organised into three sections:

1. **Section 1: Requirements decomposition process:** Questions about how engineers perform their requirements decomposition, challenges and limitations encountered during the process, completeness of the process

2. **Section 2: Quality of requirements decomposition:** Two questions regarding on how to discern quality of requirements, and how limitations on resources affect the process and artefacts.

3. **Experience with generative AI:** Questions about how the participants are using genAI in their daily life, steps they take when they are not satisfied with an answer, how genAI has affected their general workflow, and functionality prospects of a requirements decomposition tool.

The guide was designed following the principles of McGrath et al. [70]. Probes such as "Can you elaborate on that" were included to encourage extensive responses. The interview guide was designed to follow an hourglass structure: it began with a broad, open-ended question about the participant's overall approach to requirements decomposition, progressed to more focused questions, e.g., specific challenges and quality attributes, and concluded with an open-ended question inviting any remaining reflections [68]. This structure helps establish rapport early and ensures that participants have the opportunity to share contextual insights before narrowing to specific topics [70]. Furthermore, the interview guide and hourglass structure were used to maintain consistency while still allowing participants to steer the conversation towards relevant experiences. The full interview guide can be found in Appendix B.

5.3 Participants

Since this thesis was written in collaboration with Volvo Cars, participants were recruited through convenience sampling. Potential participants were either directly contacted by a contact person or were notified through a message in a company wide messaging platform. The selection criteria, to ensure relevance, were that participants should be engineers that are either actively doing requirements decomposition or have done it extensively in the past. All participants were informed about the purpose of the study, the voluntary nature of their participation, and their right to withdraw at any time. None of the interviewees withdrew post-interview participation. Interviews were held until saturation was reached as no new information was gained from interviewing new participants. Saturation was met when interviewees no longer produced new codes relevant to RQ1/ RQ2 [71].

5.4 Procedure

Each interview followed a consistent structure:

1. **Pre-interview communication:** Participants received an information sheet by email by the time of booking the scheduled interview. The sheet explained the study's purpose, the planned duration, how data would be handled, and contact details of the researchers and supervisors. A consent form was also sent, that was later filled out in the form as a Microsoft Forms at the time of interview.

2. **Start of interview:** The session began by building rapport with the participant and then with a standardised video introduction (ca. 3-minutes) that reiterated the information given out in the pre-interview communication step. This was done, in case the gap of time between the participant reading through the information sheet and the date of the interview was considerable. The recording of the interview started by filling out the consent form.

3. **Conducting the Interview:** One researcher led the interview, while the second took notes and managed the recording. This division allowed the lead interviewer to focus on active listening and probing. The interview followed the guide but allowed the participants to steer the conversation when they introduced related experiences or examples. As soon as the interview-leading researcher finished the guide's questions and potential follow up questions, the note-taking researcher also had the chance of asking questions to the participant. This was done to support active listening and reduce the risk of missed follow-ups.

4. **Post-interview:** Immediately after the interview, the researchers noted key impressions and deliberated. The recordings were stored on a secure, encrypted platform, accessible only to the research team and the participant until the time of deletion. Finally, transcripts were corrected and validated by participants, and results were reported in anonymised and aggregated form to reduce evaluation apprehension.

5.5 Data Analysis

Interview transcripts were analysed using template analysis with a hybrid deductive-inductive coding strategy to address RQ1 and inform design choices for RQ2 [68]. The initial coding template was derived from (i) the research questions, (ii) prior literature on requirements decomposition and LLMs, and (iii) informal preliminary meetings with engineers in the study context. An initial subset of transcripts was coded to calibrate the template. During this calibration, code definitions were refined and additional codes were introduced when relevant concepts were not captured by the initial template.

The resulting template comprised four code groups: (1) benefits of using LLMs as a requirements decomposition assistant, (2) limitations of using LLMs as a requirements decomposition assistant, (3) artefact design expectations, and (4) decomposition process context. After four transcripts, the calibration was complete. Therefore, the template was stabilised and applied to the remaining transcripts.

Coded segments were summarised using tabulation, where rows represent codes and columns represent participants. This structure supports a transparent chain of evidence from raw statements to themes and subsequent DOs [68]. To describe recurrence across the interview material, coded excerpts were tabulated by code and participant. For the descriptive counts, a code was counted once for a participant when at least one relevant excerpt from that participant's transcript was assigned to the code, regardless of how many times the issue was mentioned. These participant-level counts were used to distinguish isolated observations from issues raised across several interviews and to support a transparent chain of evidence from coded statements to themes and design objectives. The interpretation of each theme also considered the content, context, and relevance of the coded statements, rather than relying on counts alone.

One researcher conducted the primary coding of all transcripts. The second researcher reviewed the coding, and disagreements were resolved through discussion. When disagreements indicated ambiguity in the codebook, code descriptions and boundaries were refined to improve consistency.

While the initial template was deductively derived, it was refined iteratively during early coding to better reflect the interview data. For example, an initial code `[BEN-Speed]` was revised to `[BEN-COG]` because participants primarily described cognitive offloading, such as reduced effort for phrasing and exploratory work, rather than quantifiable time savings, which were outside the evaluation scope.

5.6 Interview Results

The following results section reports how the interview findings manifested as cross-participant themes and explains their relevance to requirements decomposition and artefact design. Overall, a single round of interviews was conducted with 12 participants in total. The first interview was conducted on the 3rd of March, whereas the final one was concluded on the 18th of March. All interviews were scheduled by the participants through a booking platform.

The results are presented in two steps before RQ1 is addressed explicitly in the following section. First, participant demographics and recurring process-level pain points are presented to describe the current requirements decomposition context. Second, artefact design expectations are presented to show how the interviews informed the later DOs in Section 5.8.

Participation Demographics and Context

The 12 participants represented a diverse range of expertise within Volvo Cars, as seen in Table 4.1. Notably, 66% (8 out of 12) already utilised genAI in their daily workflows, which could include activities outside requirements decomposition, such as code generation, while 33% (4 out of 12) did not. Furthermore, each hierarchy level was represented by at least one participant, with strong representation in the first three levels, as shown in Table 5.1. The total coverage count exceeds the number of participants because some participants worked across multiple levels within the hierarchy.

Table 5.1: Hierarchy Levels and genAI Daily Usage Represented by Participants.

Hierarchy Level	**Count**	**genAI Usage**	**Count**
Customer Function Level	7	genAI Users	8
Product Functionality Level	6	Non-genAI Users	4
Product Capability Level	7		
Group of Components	2		
(Software) Component Level	3		

General Problem Identification

To support artefact scoping and design, recurring pain points in the manual requirements decomposition process were compiled. The themes in Table 5.2 describe the baseline manual decomposition process and are used as contextual input for deriving artefact requirements for RQ2, rather than as LLM-specific limitations.

Table 5.2: Baseline Pain Points in the Manual Requirements Decomposition Process.

Theme	**Description**	**Quote**
Level Ambiguity	Difficulty in distinguishing between hierarchy levels due to a lack of clear expectations for each specific level.	"[...] we don't really have these clear guidelines and that's basically why our requirement today are refined in different ways for different features." - P2
Detail Consistency	Inconsistent granularity in requirements, ranging from overly abstract to excessively precise technical metrics.	"What do they do with one which says it should feel nice and this other requirement saying that the acceleration should be 0.3." - P11

Table 5.2: Baseline Pain Points in the Manual Requirements Decomposition Process. Continued.

Theme	Description	Quote
Methodological Gap	Lack of formal guidelines for the decomposition process.	"There are these [requirement software] rules that are not rules, but guidelines that we try to follow. But these are not strictly related to requirement decomposition." - P5
Architecture Reliance	Some requirements decomposition is heavily dependent on the individual's architectural knowledge rather than standardised procedures.	"[...] refer to the architecture to see which the components the structures [...] it should be broken down into." - P4
External Retrieval	The necessity to verify completeness by looking up data in sources external to the requirements software database.	"We don't have exactly the interfaces connected to the requirements." - P7
Downstream Verification	Completeness cannot be verified during the process, but is validated downstream by the requirement receiver.	"[handshake meetings are] definitely like the main quality check you could say, but of course we review it internally first." - P3
Informal Reviews	Absence of a formal review process, leading to requirement quality that is highly dependent on specific team cultures.	"I guess we will [find] out when testing the function if something doesn't work." - P5

The following process-context codes were used to document how decomposition is currently performed. These codes primarily deepened the researchers' understanding of the baseline process and were not treated as direct artefact requirements.

- **[PROC-COMPL] (Completeness Validation):** How teams currently ensure a decomposition is sufficient, primarily through "handshake meetings" and "internal reviews" with stakeholders.
- **[PROC-TEAM] (Team Culture):** Informal practices and local team cultures where engineers rely on "cross-checking with colleagues" rather than formal written guidelines.
- **[PROC-DOC] (Documentation Standards):** Notes the lack of unified

standards, with participants mentioning they "don't really follow Volvo Car's internal guidelines" or that guidelines vary wildly between different function areas.

Artefact Design Expectations

These codes represent specific functional expectations for the proposed LLM-assisted tool which were raised by interviewees.

- **[DES-SIM] (Historical Context):** The ability to check for previously released requirements that are related to the current decomposition.
- **[DES-RAT] (Rationale and Explanation):** The need for the LLM to provide the rationale of its logic.
- **[DES-VIS] (Visual Outputs):** A requirement for the tool to support analysing or creating visual outputs related to requirements decomposition, such as use case diagrams, UML-style diagrams, logical or functional flowcharts, and requirement relationship graphs. `[DES-VIS]` was coded as a recurrent expectation; however, graph-based artefact generation was outside the artefact scope and is reported as a future research direction.
- **[DES-INT] (Tool Integration):** The requirement for the artefact to be integrated directly into existing official tools or a web interface.
- **[DES-COLL] (Human-in-the-Loop):** Refers to statements emphasising that LLM-generated outputs should remain subject to human review and validation rather than being used autonomously.

5.7 Answering RQ1: Benefits and Limitations of LLM Assisted Requirements Decomposition

This section presents the results concerning RQ1:
What are the perceived benefits and limitations of using LLMs for requirements decomposition in the automotive industry?.
The findings are based on the thematic analysis of 12 semi-structured interviews with engineers at Volvo Cars.

Perceived Benefits of AI-Assisted Decomposition

The analysis identified three primary themes regarding the benefits of integrating LLMs into the requirements engineering workflow: decreased cognitive load, assistive support, and quality enhancements. These are reported in Table 5.3.

Table 5.3: Thematic Codes for LLM-Assisted Benefits.

Code	Definition	Mentions	Quote
[BEN-COG]	Reduction of cognitive load, such as exploratory work, phrasing, and information gathering.	11/12	"it has a huge effect on my productivity. [...] I use LLM in all the tasks I'm doing like I think I would say in all tasks I'm doing which which is everything [...]" - P3
[BEN-ASSIST]	Use of genAI as a collaborative or a non-intrusive brainstorming partner.	9/12	"Extra pair of eyes [...] without me having to disturb someone." - P1
[BEN-QUAL]	Improvement in requirement syntax, clarity, and consistency.	7/12	"Formatting properly done according to the guidelines, that is generally good enough." - P11

Cognitive offloading during decomposition work [BEN-COG]. This theme captures participants' perceptions that LLMs reduce the mental effort associated with decomposition-related activities such as exploratory information gathering, phrasing, and refinement of requirement text. In requirements decomposition, this matters because engineers must translate intent across abstraction levels while preserving clarity and traceability, which is cognitively demanding even before allocation decisions are finalised [21, 72]. For example, P3 described a substantial productivity impact and reported using an LLM across nearly all tasks. In addition, P9 emphasised reduced effort for phrasing, noting that wording which previously required several minutes can now be generated immediately, and P10 described the tool as enabling rapid learning when encountering new topics.

LLMs as an assistive companion in decomposition [BEN-ASSIST]. This theme reflects how participants used LLMs as a non-intrusive assistant for quick checks and brainstorming without the coordination overhead of contacting colleagues. This matters because requirements decomposition is often iterative and involves frequent micro-validations (e.g., checking whether wording still preserves intent) prior to formal review points [2]. The theme was mentioned by 9/12 participants. For instance, P1 described the LLM as an "extra pair of eyes" that supports verification without "having to disturb someone," and P3 framed the interaction as working with a "buddy" rather than outsourcing a task.

Quality improvements in requirement formulation [BEN-QUAL]. This theme revolves around participants' perceptions that LLMs can improve requirement quality through clearer phrasing, more consistent wording, and better adherence to writing conventions or internal guidelines. This matters because ambiguity or inconsistent granularity at higher levels can propagate downstream, increasing the

likelihood of mismatched interpretations across the hierarchy [2, 72]. For example, P11 highlighted guideline-conformant formatting as a meaningful improvement, while P1 used LLMs to "double check" whether a requirement "makes sense" and P6 emphasised the value of producing language that is "understandable by all."

Perceived Limitations and Risks of LLM Integration

Despite the perceived benefits, participants identified critical limitations, particularly regarding technical accuracy and the specific context of the automotive hierarchy. Participants often discussed LLM-related limitations together with existing decomposition challenges, as LLM outputs were expected to reflect patterns present in historical requirements. The results are presented in Table 5.4.

Table 5.4: Thematic Codes for LLM Limitations and Risks.

Code	Definition	Mentions	Quotes
[LIM-LEVELS]	Difficulty navigating or maintaining correct abstraction levels.	5/12	"[...] how technical should we go, especially on the customer [level requirements], you're not supposed to define any technical solutions [...]" - P10
[LIM-MISS]	Risk of overlooking critical technical or safety details.	6/12	"you can't have generic guidelines just of like do you use the right keywords and 'Is it testable?' That is insufficient as as a checklist." - P11
[LIM-CONT]	Lack of deep domain-specific logic or team culture knowledge.	8/12	"Passable, but [...] maybe missing some important things." - P6

Abstraction Level Challenges [LIM-LEVELS]. This theme captures concerns that LLM-generated outputs may not consistently align with the intended abstraction level when decomposing requirements. In requirements decomposition, this matters because automotive requirements are structured across multiple hierarchical levels, and mismatches between abstraction levels can lead to inconsistencies and weakened traceability across the hierarchy [2, 20, 72]. A significant risk identified is that current requirements tend to fluctuate between abstraction levels, sometimes going "a bit too low" or "a bit too high" (P1). P10 warned that even if a high-quality requirement is written, it may still be at the "wrong level" for the intended component.

Risk of Overlooking Details [LIM-MISS]. This theme reflects concerns that LLM-assisted decomposition may omit important technical, safety, or verification-relevant details, particularly when engineers rely too heavily on generated sugges-

tions. In decomposition workflows, this matters because requirements are expected to be sufficiently precise, testable, and complete, and missing details can propagate downstream into implementation and verification, increasing the risk of late rework or defects [20, 72]. P3 argued that without deep personal domain knowledge, the output of an LLM would not be of "the same quality". P6 specifically noted the LLM may not be "interpreting the input data correctly," which could lead to missed safety or technical requirements.

Lack of Domain Context [LIM-CONT]. This theme captures the limitation that LLMs may generate plausible but overly generalised decompositions that do not reflect local team practices, feature-specific constraints, or tacit organisational knowledge. In requirements decomposition, this matters because domain context and communication across organisational boundaries play a critical role in aligning requirements across abstraction levels, and gaps in such context are a known source of inconsistencies in automotive requirements engineering [2, 3, 72]. P4 observed that LLMs often provide a "generalised software decomposition" which is insufficient for specific automotive applications. P6 emphasised that the "AI cannot actually know everything from our function," making it a "passable" but often incomplete tool for complex logic.

5.8 Design Objectives

Following the identification of the primary challenges in the requirements decomposition process at Volvo Cars, a set of DOs were created. They represent the desired properties of the artefact and serve as evaluation criteria. The DOs below operationalise the interview themes into properties that the artefact must satisfy. Table 5.5 provides traceability from baseline pain points presented in Table 5.2, interview thematic codes presented in Table 5.3 and Table 5.4, and artefact design expectations presented in Section 5.6 to each objective.

Table 5.5: Traceability from Interview Results to Design Objectives.

DO	Pain Points	Thematic Codes
DO1	Cognitive Reduction	BEN-COG
DO2	Methodological Gap	BEN-QUAL
DO3	Architecture Reliance, External Retrieval	DES-SIM, LIM-CONT
DO4	Level Ambiguity, Detail Consistency	LIM-LEVELS
DO5	Architecture Reliance, Detail Consistency	DES-RAT, LIM-MISS
DO6	Downstream Verification, Informal Reviews	BEN-ASSIST, DES-COLL
DO7	External Retrieval	DES-INT

- **DO1: Cognitive Reduction.** The artefact must reduce the cognitive load

associated with repetitive phrasing and exploratory work, e.g., offering different perspectives, effectively addressing *[BEN-COG]*.

- **DO2: Contextual Grounding in Technical Standards.** The artefact must incorporate domain-specific standards and internal Volvo guidelines (e.g., Safety Standards, NASA guidelines) to ensure that the generated requirements are technically valid and compliant. This objective addresses the *Methodological Gap* and the need for *[BEN-QUAL]*.

- **DO3: Integration of Released Project Knowledge.** The artefact should provide engineers with access to relevant approved-for-release requirements and architectural context. This objective aims to mitigate the identified *Architecture Reliance*, the need for *External Retrieval* and *[DES-SIM]*.

- **DO4: Maintenance of Structural Hierarchy and Abstraction.** The artefact must assist in maintaining the correct level of granularity when decomposing requirements from Customer Level Functions down to lower levels. This objective is a direct response to the identified *Level Ambiguity* and *Detail Consistency* issues and *[LIM-LEVELS]*.

- **DO5: Keeping Intent and Purpose Intact.** The artefact must consider the source requirement's purpose and intent to assist the system engineer to perform requirements decomposition. This objective addresses *[DES-RAT]*, the need for the system engineer to understand the agent's rationale, and the need to avoid generic requirements or missing details required to perform the decomposition, *[LIM-MISS]*.

- **DO6: Facilitation of Human-in-the-Loop Validation.** Recognising that completeness cannot be verified autonomously, the artefact should act as a "draft generator" or "brainstorming partner" rather than a fully autonomous system. It must provide clear rationales for its outputs to allow for efficient human review, addressing the *Downstream Verification*, *Informal Reviews*, and *[DES-COLL]*.

- **DO7: Assistant Integration in Current Requirements Software.** The artefact must be integrated into the current requirements software engineers are using when performing the decomposition process. This can lessen the burden of *External Retrieval* by reducing the need to switch context windows and fulfils *[DES-INT]*.

6

Artefact Design

This chapter presents the design of the proposed requirements decomposition artefact. It first introduces the overall design positioning and guiding DOs, followed by the iterative development process through which the artefact evolved during the DSR cycle. The chapter concludes by presenting the information flow within the final artefact, integrating the mechanisms introduced throughout the iterations.

6.1 Design Overview and Positioning

The proposed artefact is an LLM-based requirements decomposition workflow embedded in an existing multi-agent environment, as described in Section 4.4. Within this environment, an orchestrator coordinates specialised agents and dynamically selects appropriate workflows based on the task and interaction context. The requirements decomposition workflow is integrated as one such callable workflow, equipped with a defined output contract that specifies its expected inputs, outputs, and execution behaviour. This enables the orchestrator to invoke the decomposition workflow when a user query or task aligns with requirements decomposition. This means that the surrounding agent ecosystem provides the context in which the workflow is invoked, while the internal stages of requirements decomposition are defined by the artefact itself.

The workflow itself supports engineers in refining higher-level requirements into lower-level candidate specifications through staged contextual grounding and human-guided decomposition support. Its design was guided by the DOs introduced in Section 5.8 and a mapping of each DO to specific artefact mechanisms and their intended contribution can be seen in Table 6.1. Rather than treating decomposition as a single prompt-response task, the artefact was designed as a staged workflow in which relevant context is gathered, validated by users, and organised before final generation.

Table 6.1: Mapping between Design Objectives and Artefact Mechanisms.

Design Objective	Artefact mechanism(s)	Intended contribution
DO1: Cognitive reduction	Structured workflow; intermediate summaries; generated candidate decomposition	Reduces the need to start decomposition from a blank page by organising context and producing an initial draft for expert review.
DO2: Contextual grounding in technical standards	RAG Agent; standards and guideline retrieval; user validation of retrieved principles	Provides standards-related context that engineers can inspect before final decomposition.
DO3: Integration of released project knowledge	KG-RAG Agent; semantic seed retrieval; graph expansion; historical decomposition retrieval	Supports comparison with previous decomposition patterns by retrieving related historical examples.
DO4: Maintenance of structural hierarchy and abstraction	Hierarchy rules; source-target level guidance; PSE-linked hierarchy information; structured prompt orchestration	Supports alignment between the generated output and the intended abstraction transition.
DO5: Keeping intent and purpose intact	PSE Summary Agent; final prompt orchestration; schema-constrained decomposition; KDPs; Observations	Uses contextual summaries and structured explanatory outputs to preserve parent intent and expose issues for review.
DO6: Facilitation of human-in-the-loop validation	Validation checkpoints after PSE, RAG, KG-RAG, and final decomposition outputs	Keeps contextual inclusion and final acceptance dependent on user inspection rather than automatic generation alone.
DO7: Assistant integration in current RMS	Workflow embedded in existing assistant environment; retrieval and generation within one decomposition workflow	Reduces context-switching by bringing contextual information and generated suggestions into a single workflow.

6.2 Design Iterations

The artefact was developed through five iterations, with each iteration incrementally extending the previous version through additional design mechanisms in response to limitations observed in the previous version. The mechanisms were later incorporated into the final orchestrated workflow presented in Section 6.3. A central design principle throughout the artefact development was that contextual grounding mechanisms should remain configurable by the engineer. Rather than enforcing mandatory inclusion of retrieved standards, historical decompositions, or generated contextual summaries, the workflow allows contextual inputs to be reviewed, revised, selectively incorporated, or excluded before downstream generation.

While embedded in a multi-agent environment, the decomposition workflow follows a staged process for constructing the final prompt. It consists of a fixed sequence of agents, each producing intermediate outputs that may be incorporated together

with rule-based elements such as hierarchy guidance. The stages are executed in a predefined order, where intermediate outputs are progressively combined during prompt construction.

Although the workflow follows a constrained staged execution process, it adopts several multi-agent design principles. Each agent is implemented as an independent LLM-based unit with defined input expectations and schema-constrained outputs, ensuring consistent interaction between stages. The final prompt is therefore constructed incrementally from selected intermediate outputs and predefined structural rules.

Following the formative evaluation strategy described in Section 4.4, this section presents the iterations in terms of the limitations they revealed, the design changes they motivated, and the remaining issues that shaped following refinements.

Each iteration concludes with a summary of the primary design insights and limitations identified during the formative evaluation, contributing to the analysis of RQ2.

Iteration 1: Workflow Integration and Artefact Scoping

The first iteration focused on integrating a requirements decomposition workflow into the existing multi-agent assistant environment. Rather than modifying existing agents, the decomposition workflow was implemented as an isolated pipeline within the broader infrastructure.

This design decision ensured artefact development stayed independent while preserving the stability of the surrounding assistant ecosystem. The iteration therefore established a viable integration strategy and demonstrated the feasibility of incorporating LLM-assisted decomposition into the existing workflow environment.

At this stage, the workflow relied on direct LLM prompting without additional contextual grounding or hierarchy-specific constraints, serving as a baseline for subsequent refinements. Despite its simplicity, the workflow demonstrated that LLMs were capable of generating structurally plausible lower-level candidate requirements from higher-level stakeholder input.

Lessons Learned: Formative evaluation revealed important limitations related to contextual grounding and abstraction control. The decomposition mostly resembled an atomic subdivision of the input requirement rather than a refinement adapted to the intended system context. In particular, the workflow lacked mechanisms for resolving decomposition context, intended abstraction level, and technical granularity, occasionally resulting in unsupported assumptions or inconsistent detail in the generated requirements.

These findings indicated that requirements decomposition could not reliably be treated as a purely semantic or linguistic transformation task, motivating the introduction of contextual grounding mechanisms in subsequent iterations.

Iteration 2: Grounding through Standards

To address the lack of contextual grounding identified in iteration 1, a RAG agent was introduced to retrieve semantically relevant standards and internal guidelines at runtime. Rather than directly injecting retrieved text into the final prompt, the retrieved chunks were processed through a structured prompting workflow designed to extract decomposition-relevant principles.

The RAG agent therefore acted as an intermediate contextualisation step between retrieval and generation. However, initial formative evaluation with engineers revealed limitations in the retrieval behaviour. Similarity-based retrieval often failed to capture the broader context of the source document, resulting in chunks that appeared relevant in isolation but originated from standards or documents that were not applicable to the decomposition task.

To improve retrieval precision, document-level context was incorporated into each chunk representation prior to embedding. This was achieved by prepending a short description of the source document to each chunk. This allowed the retrieval process to better prioritise relevant standards and guidelines, rather than relying solely on local semantic similarity.

The structured prompting workflow further included mechanisms for handling incomplete or vague retrieval results, such as fallback behaviour and filtering instructions, in order to improve robustness and reduce irrelevant contextual grounding. After principle generation, the engineer validated the extracted principles before they were incorporated into the final prompt construction process. This maintained human oversight while allowing contextual grounding to be adapted to the specific decomposition task. An overview of the RAG Agent workflow is shown in Figure 6.1.

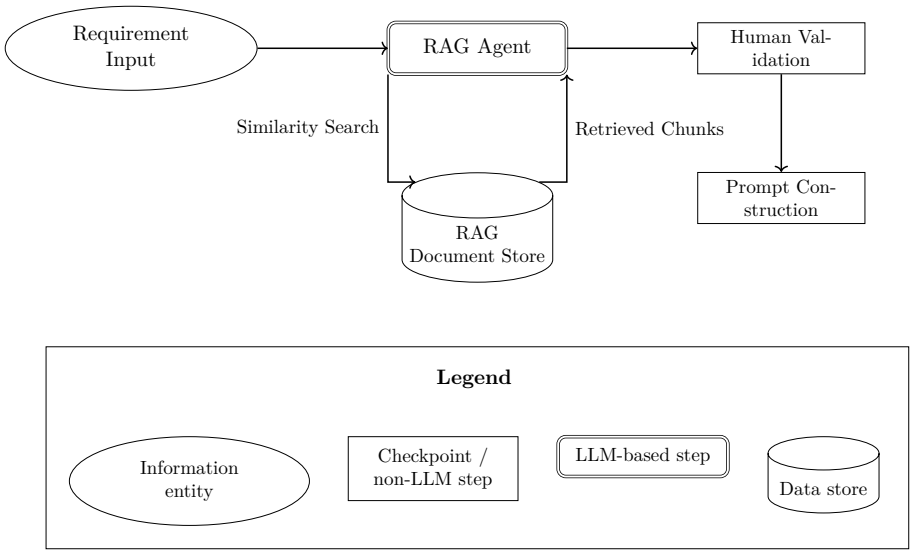

Figure 6.1: Overview of the RAG Agent Workflow.

Lessons Learned: The introduction of retrieval-based grounding improved contex-

tual alignment by incorporating standards-oriented guidance and engineering constraints into the decomposition workflow. However, formative evaluation revealed that retrieved standards and guidelines were sometimes too generic or abstract to fully support practical decomposition decisions. While standards could provide relevant principles, terminology, and constraints, they often lacked concrete guidance for how a specific requirement should be decomposed in a particular engineering context. As a result, engineers still needed to validate whether the generated principles were sufficiently relevant and applicable to the current decomposition task.

Iteration 3: Historical Decomposition Reuse through KG-RAG

Whereas previous iterations focused on grounding generation in formal documentation, this iteration aimed to expose practical decomposition experience embedded in historical engineering work. The objective was to address the lack of organisation-specific engineering context identified in previous iterations by enabling reuse of previously developed decomposition patterns and engineering decisions.

Unlike standards and guidelines, the primary value of historical requirements was not the individual requirement text itself, but the associated decomposition structures created during previous decompositions. In this study, historical decompositions refer to previously released parent-child requirement refinement structures stored within the RMS. Retrieving isolated semantically similar requirements from a vector store was therefore insufficient, since similarity search alone could not recover how these requirements had historically been decomposed across abstraction levels.

A knowledge graph mirroring the RMS was used to reconstruct historical decomposition relationships within a release context. Starting from a semantically similar requirement identified through similarity search within a requirements store, graph traversal was used to retrieve connected lower-level refinement structures associated with the same decomposition chain.

Instead of incorporating retrieved decompositions directly into generation, the workflow presented retrieved decomposition structures to the engineer for contextual selection. The selected decompositions were then processed through a structured prompting workflow to extract decomposition-relevant contextual patterns and keywords for downstream prompt construction. An overview of the KG-RAG workflow is shown in Figure 6.2.

Lessons Learned: Incorporating historical decompositions improved contextual alignment and completeness by enabling reuse of organisation-specific decomposition patterns. By surfacing historical decomposition structures, the workflow exposed system-specific context relevant to the decomposition task.

However, formative evaluation revealed important limitations. Retrieved decompositions did not necessarily reflect universally applicable engineering practices and occasionally introduced contextual noise. The approach was also less suitable for

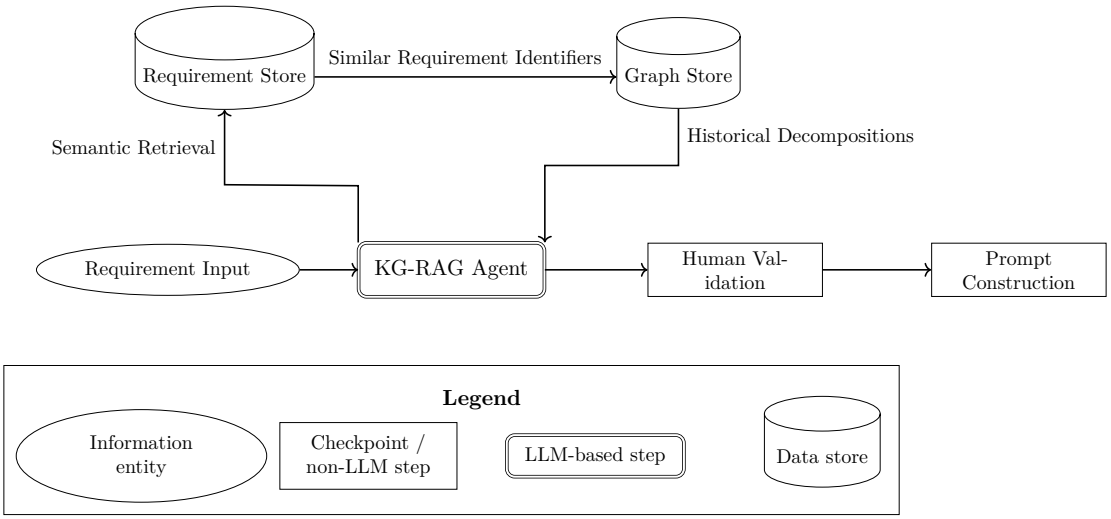

Figure 6.2: Overview of the KG-RAG Agent Workflow.

novel or highly specialised requirements, as it tended to bias the model toward previously used solutions. In addition, reuse of historical decompositions may propagate existing limitations or inconsistencies into new decompositions.

Furthermore, historical decompositions were not always directly interpretable. They often required contextual understanding of their original application, increasing the effort for engineers to assess their relevance. This can introduce additional cognitive load, partially counteracting the intended benefit of supporting the decomposition process.

Iteration 4: PSE-Based Grounding and Hierarchical Rules

While previous iterations improved contextual grounding through standards and historical decompositions, formative evaluation revealed remaining issues related to decomposition context and abstraction-level consistency. Generated decompositions occasionally exhibited misalignment with subsystem responsibilities and inconsistent technical granularity, particularly when multiple contextual sources were combined.

To address the lack of structural system context, the fourth iteration introduced PSE-based grounding within the decomposition workflow. PSE-linked information was incorporated to provide system-level context and help align generated sub-requirements with their intended subsystem and hierarchy position.

Initial experiments attempted to incorporate raw PSE-linked data directly into the prompt context. However, the retrieved information often consisted of large volumes of metadata and attached requirements, represented in structured schema-like formats. For complex PSE structures, the resulting context became difficult to interpret consistently and could become impractical for downstream prompt construction.

To address these limitations, a PSE Summary Agent was introduced to transform retrieved PSE-linked information into a semantically interpretable system-context

summary suitable for downstream prompt construction. A human validation checkpoint was additionally integrated at this stage, allowing engineers to verify, revise, or discard the generated summary before downstream processing. The resulting workflow is illustrated in Figure 6.3.

In parallel, hierarchical decomposition rules were introduced to improve abstraction level consistency during generation. Earlier iterations relied primarily on RMS-derived hierarchy associations, which proved insufficiently precise in certain contexts. Explicit source- and target-level descriptions derived from PSE-linked hierarchy information were therefore incorporated into the final prompt to guide the intended abstraction transition.

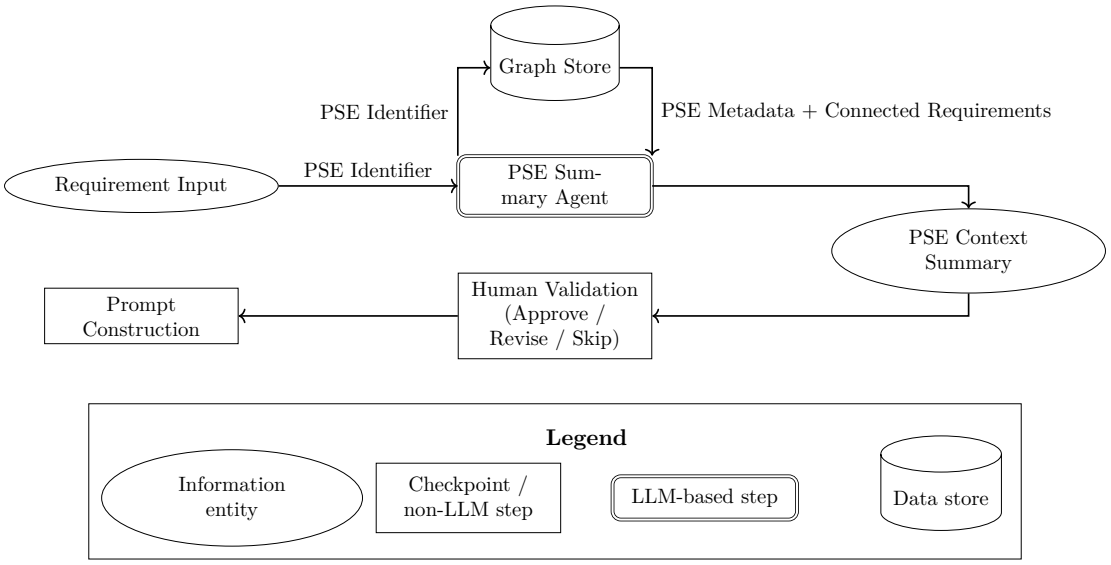

Figure 6.3: Overview of the PSE-based Context Integration Process.

Iteration 5: Structured Prompt Orchestration and Controlled Generation

While previous iterations progressively introduced additional grounding mechanisms, the integration and prioritisation of heterogeneous contextual sources remained insufficiently structured. In particular, combining standards, historical decompositions, PSE-derived context, hierarchical rules, and user guidance occasionally produced conflicting signals and inconsistent abstraction boundaries during decomposition.

As contextual complexity increased across iterations, prompt construction became increasingly difficult to manage consistently. The fifth iteration therefore formalised the orchestration process into a dedicated prompt construction stage responsible for consolidating and prioritising contextual evidence before generation. This orchestration approach was informed by the prompt engineering principles discussed in Section 2.5, particularly the separation of prompt components, contextual ordering, applicability filtering, and structured output specification mechanisms.

The resulting structured prompt template combined the original requirement and user guidance with PSE-derived context, retrieved standards, historical decomposition examples, and hierarchical decomposition rules.

To improve abstraction consistency and contextual alignment, the orchestration stage introduced explicit prioritisation and ordering strategies. User input, structural constraints and standards-based grounding were prioritised before historical decomposition examples to reduce ambiguity and limit overfitting towards retrieved refinement patterns. The orchestration stage additionally standardised how intermediate agent outputs were represented and incorporated into downstream prompt construction. Additional implementation-specific prompt engineering techniques and generation parameters are provided in Appendix A and Appendix D, respectively.

The final decomposition stage operated under explicit structural and contextual constraints derived from the orchestrated prompt. In addition to generated candidate sub-requirements, the workflow produced supporting rationale, observations, and contextual information intended to improve interpretability during engineer review. Conservative generation settings were additionally introduced to improve consistency and controllability across workflow executions.

Lessons Learned: Formalising the orchestration process improved decomposition consistency and enabled more reliable integration of heterogeneous contextual sources. The structured rationale and observation outputs additionally improved transparency during human review by exposing assumptions, ambiguities, and contextual dependencies identified during generation.

At the same time, iterative refinement of prompt composition and contextual prioritisation highlighted the sensitivity of LLM-based decomposition workflows to orchestration strategy and contextual ordering.

6.3 Final Artefact Architecture

Together, the five design iterations transformed the artefact from a standalone LLM prompting workflow into a structured HITL decomposition architecture integrating contextual retrieval, hierarchical guidance, orchestration, and constrained generation.

The final architecture separates the workflow into distinct stages with different responsibilities. Retrieval agents resolve standards, historical decompositions, and project-specific structural information from heterogeneous engineering knowledge sources, while intermediate interpretation stages transform retrieved information into contextual representations suitable for downstream reasoning and prompt construction.

Rather than exposing retrieved information directly to generation, the workflow incorporates validation checkpoints throughout the process, allowing engineers to review, revise, discard, or selectively include contextual information before down-

stream processing. This maintains the role of the artefact as a decision-support workflow rather than an autonomous decomposition system.

A dedicated orchestration stage consolidates validated contextual inputs into a structured reasoning context prior to generation. The final decomposition stage then operates under explicit structural and contextual constraints derived from the orchestrated prompt, producing both candidate sub-requirements and supporting contextual artefacts intended to improve interpretability during engineer review. While earlier iterations relied on GPT-4.1 for the final generation step, the final implementation uses GPT-5.4. This change was motivated by improved consistency and control observed during later testing.

An overview of the information flow within the final artefact is shown in Figure 6.4. The figure shows how information flows through the artefact, not the exact order in which the steps are run. In the implementation, the contextual agents and hierarchy rules are run sequentially, but their validated outputs are only brought together in the structured prompt orchestration stage. The architecture formed the basis for the summative evaluation presented in Chapter 7. Specific implementation parameters are provided in Appendix D.

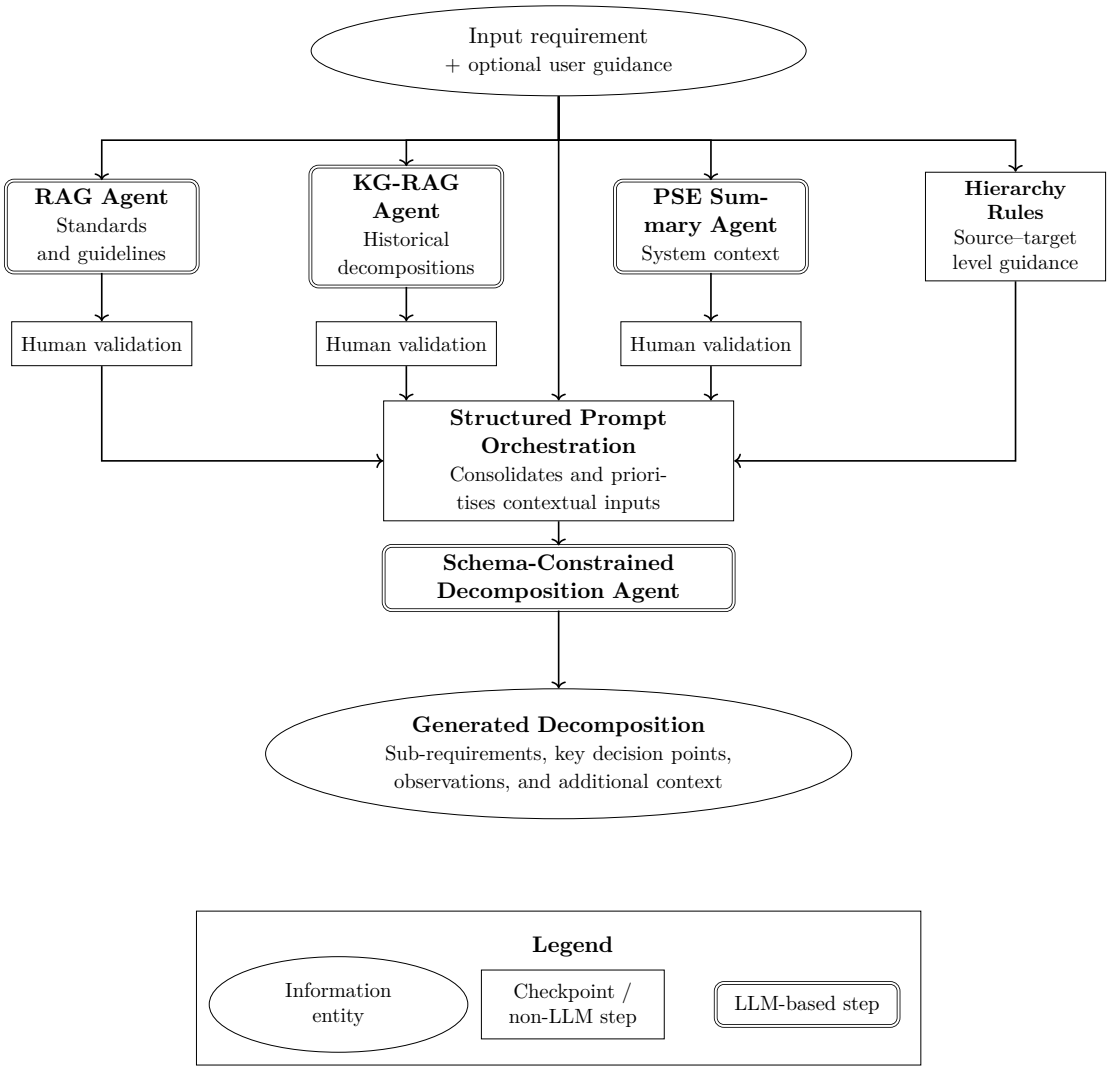

Figure 6.4: Overview of the information flow within the final artefact.

7

Final Evaluation

This chapter goes over the final evaluation and its results. First, an alignment between DOs and the evaluation is established then the procedure is presented. The second half of the chapter presents the results.

7.1 Final Evaluation Overview

Alignment Between Domain Knowledge, Design Objectives, and Evaluation

The evaluation design was constructed to preserve traceability between the industrial problem investigation, the artefact design, and the final assessment. The evaluation framework combines two sources of knowledge. First, practitioner input from the relevance cycle defined which aspects of decomposition support were important in the industrial context. This included the need to assess whether generated requirements were sufficiently complete, technically meaningful, consistent with the parent requirement, and expressed at the appropriate abstraction level. Second, RE quality attributes established by Denger [10] were used to refine these practitioner concerns into evaluation dimensions such as correctness, consistency, completeness, comprehensibility, unambiguity, modifiability, and right level of detail. In this way, the evaluation reflects both industrial relevance and theoretical grounding.

The resulting evaluation dimensions were applied to the final artefact. The intermediate workflow components were assessed according to their intended contribution to the decomposition process: the PSE Summary Agent was evaluated in terms of contextual accuracy and completeness, the RAG Agent in terms of standards-related correctness and relevance, and the KG-RAG Agent in terms of the relevance and interpretability of historical decomposition examples. Human interaction aspects, including usefulness and adoption rate, were addressed through Technology Acceptance Model (TAM) [73] related items and open-ended questions. A mapping can be seen in Table 7.1.

Table 7.1: Alignment between DOs and Evaluation Focus.

Design Objective	Evaluation focus	Evaluation phase
DO1 Cognitive Reduction	Perceived usefulness, productivity impact, and reduced effort	Phase 3 and TAM-related items
DO2 Contextual Grounding in Technical Standards	Correctness, relevance, and usefulness of retrieved standards-related principles	Phase 2 RAG Agent assessment
DO3 Integration of Released Project Knowledge	Relevance and contextual usefulness of historical decomposition examples	Phase 2 KG-RAG Agent assessment
DO4 Maintenance of Structural Hierarchy and Abstraction	Correct abstraction level, granularity, and right level of detail	Phase 1 and Phase 2 decomposition assessment
DO5 Rationale and Contextual Explanation	Usefulness of KDPs, observations, and additional contextual information	Phase 1 and 2 decomposition assessment
DO6 HITL Validation	Workflow steps	Phase 2 and Phase 3
DO7 Workflow Integration	Workflow fit, usability, and adoption conditions	Phase 3 and TAM-related items

Final Evaluation Procedure

To assess the utility and output quality of the proposed artefact in an industrial context, a structured evaluation was conducted with domain experts. The evaluation combined a controlled benchmark assessment, a context-dependent assessment using participant-selected requirements, and open-ended questions concerning perceived usefulness, risks, and improvement opportunities. The results are presented in Section 7.2.

The evaluation consisted of three phases:

- **Phase 1: Benchmark Evaluation**
  Participants were presented with a fixed input requirement and its corresponding generated decomposition. Although the workflow supports optional user guidance during decomposition, no such guidance was included in the benchmark case. Intermediate system steps were not shown. This phase enabled controlled comparison across participants, as all evaluations were based on identical inputs and outputs. The benchmark requirement was selected because it was written in customer-facing language, making the expected decomposition more product-centred rather than highly technical. This was intended to make the output easier to assess without detailed system-specific knowledge.

- **Phase 2: Contextual Evaluation**
  Participants interacted with the artefact using a requirement from their own domain. This phase included assessment of intermediate workflow outputs, such as the PSE Summary Agent, RAG Agent, and KG-RAG Agent, as well as the final generated decomposition. This allowed participants to evaluate the artefact in relation to system-specific constraints, architectural knowledge, applicable standards, and practical decomposition needs.

- **Phase 3: Open-Ended Questions**
  Participants answered open-ended questions concerning the artefact's role as a collaborative partner, the effect of reduced context-switching, perceived safety-critical risks, suggested improvements, trust in correction-based learning, and additional comments.

The three-phase design balanced comparability, contextual validity, and qualitative feedback. Phase 1 supported consistent cross-participant evaluation of general decomposition properties, such as abstraction level, consistency, and completeness. However, because the benchmark requirement was fixed, participants may not be able to assess technical correctness based on their own domain knowledge. Phase 2 addressed this limitation by allowing participants to evaluate outputs within a familiar system context, thereby supporting assessment of correctness & technical validity more accurately, and alignment with their own practice. Phase 3 complemented the structured questionnaire by capturing participants' broader reflections on workflow fit, risks, adoption conditions, and potential future improvements.

Evaluation Setup

The evaluation sessions were conducted remotely via Microsoft Teams. Each session was recorded and automatically transcribed to support analysis of qualitative responses.

Due to the prototype nature of the artefact, participants did not directly operate the system interface. Instead, the researcher executed the workflow locally while allowing participants to guide the process by selecting inputs, interpreting intermediate outputs, and deciding how to proceed at each step. This ensured consistent execution across sessions while preserving participant involvement in the decision-making process.

Evaluation Framework

To evaluate the capabilities of the proposed LLM-based requirements decomposition artefact, this study utilises the TAM [73] to evaluate tool adoption. The TAM proposes that user adoption intentions are primarily driven by two core factors: perceived usefulness and perceived ease of use. Furthermore, the extended requirements quality framework established by Denger & Olsson [10] is utilised. While traditional engineering standards such as IEEE 830 [74] focus primarily on rigid, machine-verifiable documentation constraints, Denger & Olsson [10] introduced human-centric and contextual attributes, such as *Comprehensibility* and the

Right Level of Detail to ensure that specifications are optimised for human stakeholders.

Data Collection

Quantitative data was collected using a structured questionnaire consisting of Likert-scale items on a five-point scale: “Strongly disagree”, “Disagree”, “Neither agree nor disagree”, “Agree”, and “Strongly agree”. The questionnaire was completed during the session, with responses discussed and validated directly with the participant to ensure correct interpretation.

To support empirical and theoretical traceability, the evaluation questions were mapped to requirements quality attributes adapted from Denger and Olsson [10], as detailed in Table 7.2. In addition to the quantitative questionnaire, open-ended questions were used in Phase 3 to capture qualitative feedback regarding perceived usefulness, workflow fit, safety-critical risks, suggested improvements, and trust in correction-based learning. The full questionnaire is provided in Appendix C.

Table 7.2: Mapping of Evaluation Questions to Denger & Olsson Quality Attributes.

Question Group / Handle	**Evaluation Query Variable**	**Target Quality Attribute**
Phase 1 Baseline & Phase 2 Overall	Logically consistent decomposition	Correctness & Consistency
	Correct abstraction level	Right Level of Detail
	Distinct responsibilities	Unambiguity & Modifiability
	Sufficient completeness	Completeness
	Distinct responsibilities	Right Level of Detail
	KDPs help validate/understand & Observations provide context	Comprehensibility
	Alignment with external constraints / standards	Consistency
Phase 2 Summary Agent	PSE Summary is accurate	Correctness
	PSE Summary captures relevant details	Completeness
Phase 2 RAG Agent	RAG Principles are correct	Correctness
	RAG Principles are relevant / Facilitates identifying standards	Consistency & Completeness
Phase 2 KG-RAG Agent	Historical decompositions are relevant / Facilitates finding decompositions	Consistency
	Output provides enough context	Comprehensibility

7.2 Final Evaluation Results

This section reports the results from the final evaluation round. The purpose is to present the observed quantitative response distributions and recurring qualitative themes. The results are described at the level of observed response patterns, while their implications for the artefact design and research questions are discussed in Chapter 8.

Due to the qualitative and domain-specific nature of requirements decomposition, the evaluation was conducted with a focused cohort of ten domain experts. As described, in section 4.2, eight of these participants had also taken part in the interview phase. Therefore, some evaluation participants were already familiar with the study context and problem framing. The results are visualised using count-based diverging Likert charts. Neutral responses, corresponding to score 3 ("Neither agree nor disagree"), are shown numerically but are not treated as active agreement or disagreement. Statements marked with an asterisk ($*$) denote reverse-coded questions, where disagreement indicates a positive assessment. These items were included to check whether participants perceived the generated decomposition as introducing unrelated or out-of-scope content.

Integrated System Results

As described in Section 7.1, the final evaluation was divided into two main phases. Phase 1 used a fixed benchmark requirement to assess the baseline quality of the generated decomposition under comparable conditions across participants. Phase 2 used participant-selected requirements from their own domain to assess the artefact in a more contextual and industrially realistic setting. During Phase 2, participants were guided through the workflow and answered the corresponding questions at each relevant step. For clarity, the Phase 2 results are presented by artefact component and by overall decomposition quality.

Phase 1: Benchmark Evaluation Results

Phase 1 evaluated participants' perceived logical consistency and structural quality of the generated decomposition using a fixed input requirement. No participant was familiar with the system context of the shown requirement. Therefore, participants were not presented with the question regarding validity and compliance with system design & standards. The results are shown in Figure 7.1.

The benchmark evaluation showed a generally positive response pattern for the shared input-output example. Participants tended to perceive the decomposition as logically consistent with the input requirement, at an appropriate level of abstraction, and sufficient for fulfilling the parent requirement. For the reverse-coded item, most participants disagreed that the generated sub-requirements introduced unrelated functional responsibilities. This indicates that they generally did not perceive the decomposition as expanding beyond the parent requirement. The supporting explanation elements, including KDPs and Observations, were also rated positively,

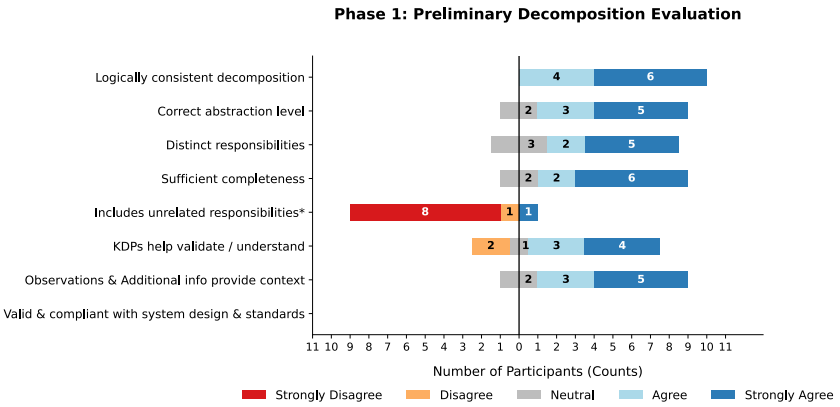

Figure 7.1: Phase 1: Benchmark Evaluation Results. Reverse-coded statements are marked with an asterisk (∗).

although with slightly weaker consensus than the core decomposition-quality dimensions.

These results should be interpreted in relation to the benchmark setup. Since all participants assessed the same generated decomposition and were unfamiliar with the specific system context, the results primarily indicate perceived plausibility and structural clarity rather than verified correctness in the original engineering context.

Phase 2: Contextual Evaluation Results

Phase 2 evaluated the artefact in a contextual setting, where participants assessed the workflow using requirements from their own domain. Unlike Phase 1, this phase included the intermediate support components used to ground and guide the final decomposition. The results are therefore presented by artefact component: the PSE Summary Agent, the RAG Agent, the KG-RAG Agent, and the final generated decomposition.

Summary Agent

The Summary Agent was evaluated with respect to the accuracy and relevance of the generated PSE summary. The results are shown in Figure 7.2.

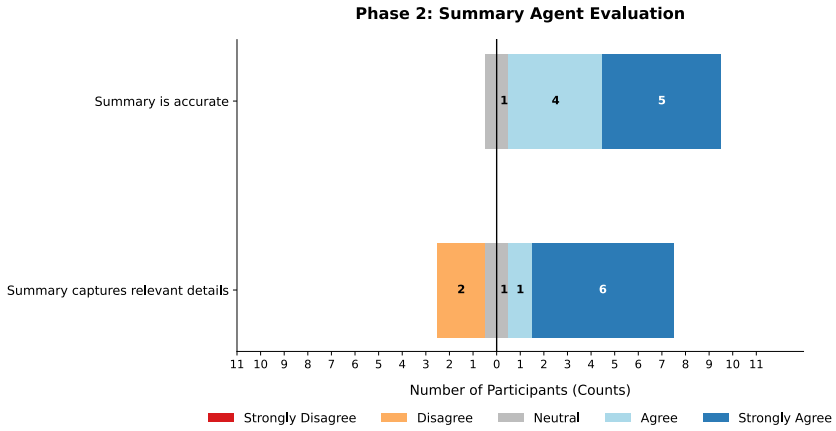

Figure 7.2: Summary Agent Assessment.

The Summary Agent received a positive assessment overall. Participants most strongly agreed that the generated PSE summary was accurate, with no disagreement on this item. The relevance of the summary was also assessed positively by most participants, although this dimension showed more variation with two participants disagreeing.

RAG Agent Results

The RAG Agent was evaluated with respect to the correctness and relevance of the suggested principles, as well as its ability to facilitate identification of relevant standards. The results are shown in Figure 7.3.

The RAG Agent results were moderately positive overall, with more neutral responses than the Summary Agent. For correctness and standards identification, six participants agreed or strongly agreed, while the remaining responses were neutral. Relevance showed the most variation, with six participants agreeing or strongly agreeing, two neutral responses, and two disagreement responses.

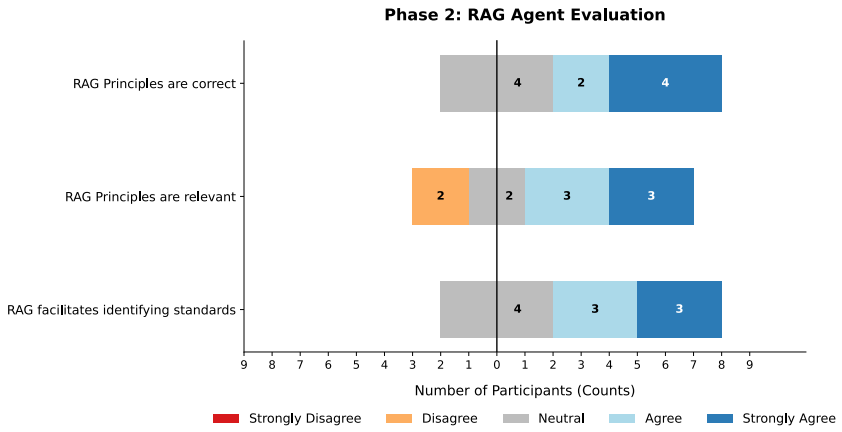

Figure 7.3: RAG Agent Regulatory Standards Mapping Evaluation.

KG-RAG Results

The KG-RAG Agent was evaluated with respect to its ability to provide useful context from historical decompositions. The results are shown in Figure 7.4.

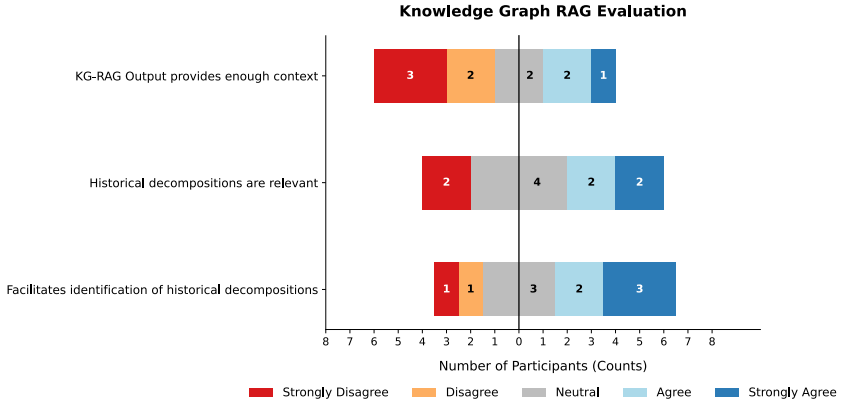

Figure 7.4: KG-RAG Context and Historical Decomposition Evaluation.

The KG-RAG results were the most mixed among the intermediate support components. The item concerning identification of relevant historical decompositions received five agreement or strong-agreement responses, alongside three neutral and two disagreement responses. The relevance of the suggested historical decompositions was more evenly distributed, with four agreement or strong-agreement responses, four neutral responses, and two strong-disagreement responses. The weakest result concerned whether the KG-RAG output provided enough context to understand the retrieved historical decomposition, where disagreement responses outnumbered agreement responses.

Decomposition Results

The final decomposition output was evaluated with respect to logical consistency, abstraction level, overlap, completeness, functional relevance, explanatory support, and technical validity. The results are shown in Figure 7.5.

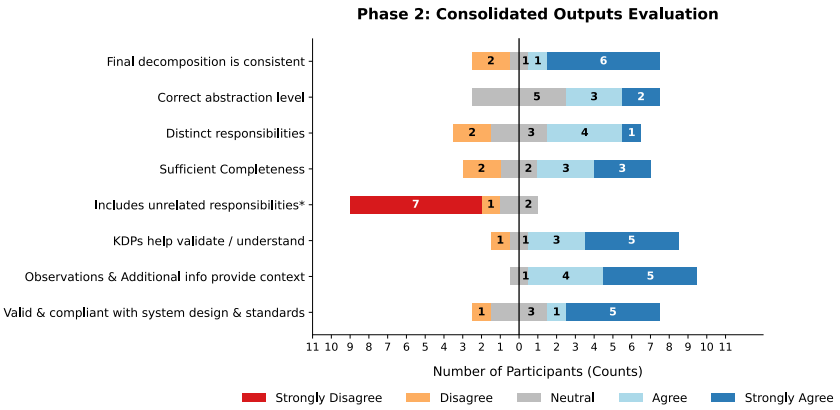

Figure 7.5: Phase 2 Decomposition Assessment. Reverse-coded Statement are marked with an asterisk (∗).

The final decomposition output received a generally positive assessment across most evaluated dimensions. Participants tended to assess the generated decomposition as logically consistent, sufficiently complete, technically valid, and not introducing unrelated functional responsibilities. The explanatory support elements, including KDPs and Observations, were also evaluated positively. The abstraction-level results showed weaker consensus, indicating more uncertainty around whether the generated sub-requirements consistently matched the intended level of detail.

TAM Results

The TAM-related questions evaluated participants' perceived intention to use the tool, willingness to recommend it, expected quality improvement, interaction intuitiveness, and perceived training effort. The results are shown in Figure 7.6.

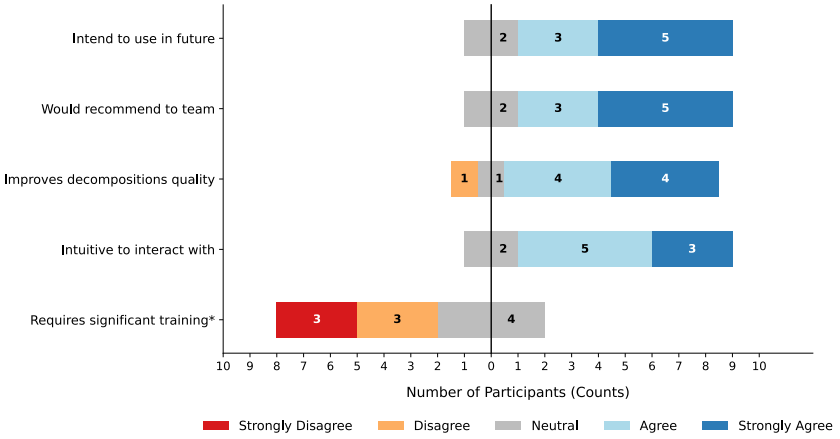

Figure 7.6: TAM Usability and Adoption Trends. Reverse-coded Statement are marked with an asterisk (∗).

The TAM-related results showed a positive response pattern for potential adoption. Eight participants agreed or strongly agreed that they would use the tool if integrated into the RMS, recommend it to colleagues, and found the interaction

intuitive. The expected quality-improvement item also received eight agreement or strong-agreement responses, with one neutral and one disagreement response. For the reverse-coded training-effort item, no participants agreed that substantial training would be required.

Phase 3: Open-Ended Questions

The open-ended questions within the final evaluation, as can be seen in Appendix C, focused on six aspects of the artefact:

- The artefact's role as a collaborative partner
- The effect of reduced context-switching
- Perceived safety-critical risks
- Suggested improvements
- Trust in correction-based learning
- Any additional comments

In this section, the participants' responses are grouped into recurring themes. The themes capture how participants described the artefact's role in the decomposition workflow, where they saw potential risks, and which improvements they considered relevant for practical use.

Theme 1: Additional Decomposition Perspectives

A recurring pattern in the open-ended responses was that participants valued the artefact for providing an additional perspective on the decomposition task. Several participants framed this as a second opinion or complementary input (P1, P13). Others described using the tool after an initial interpretation to compare and refine the decomposition (P9, P15). Participants also highlighted its ability to surface related structures and insights not readily available from existing tools (P6, P10, P12). Overall, the artefact's value was linked to comparison and reflection, enabling alternative interpretations of the decomposition task.

At the same time, this perspective was expected to remain subject to human judgement. Participants emphasised that outputs should be reviewed and adapted rather than used as a source of truth (P6, P13). P10 also expressed uncertainty regarding how context-selection decisions influence the generated output.

Theme 2: Reduced Context-Switching Through Embedded Retrieval

Most participants described the integration of retrieval and decomposition within a single workflow as useful. P10 linked the combined workflow to improved efficiency,

stating that it would make engineers "save time and be more efficient in doing the breakdown." P13 expressed a similar view, stating that having retrieval and decomposition within the same workflow would improve the process. P6 also emphasised this benefit and related it to current work practices, describing the existing RMS as slow when switching between information sources.

However, participants also highlighted limitations to this benefit. P1 emphasised that usefulness depends on how quickly the workflow can be completed, noting that multiple steps, confirmations, and loading times may reduce its practicality for lightweight tasks. P1 also pointed out that retrieved standards and historical decompositions may be excessive for simpler requirements.

The relevance of reduced context-switching also varied across participants. For example, P9 did not describe separate information search as a major issue in their current workflow.

Theme 3: Risks in Safety-Critical Use

Participants identified several potential risks related to using the artefact in production. A recurring concern was that users may rely too heavily on the generated output. P1 described this as a risk that users may begin to rely on the tool completely and "stop thinking" themselves. P5 and P6 expressed similar concerns, emphasising the need for manual review and continued human responsibility, particularly in production contexts. P15 also noted that they would not fully trust the generated output and that resulting requirements should be reviewed by responsible engineers or teams before use.

P13 raised a related issue, stating that users may copy generated text into the official requirement tool without fully reading or understanding it. P13 also noted that this behaviour is not unique to the artefact, as similar practices may occur with general-purpose LLM chatbots. In addition, P13 described that keeping the assistant separate from the production requirement tool may introduce beneficial friction that encourages users to review the output more carefully.

Other risks concerned the contextual validity of the generated decomposition. P9 emphasised that the model may lack access to recent internal updates, which could result in suggestions based on outdated practices. P12 described limitations in the tool's contextual understanding, including difficulty distinguishing between system functions, abstraction levels, and applicability of retrieved information.

P14 identified a different type of risk related to the presentation of generated output. They noted that even when the decomposition is technically correct, excessive verbosity may make it harder to interpret, as requirement texts need to be read and understood efficiently by the receiving engineer.

Overall, participants described risks related to overreliance, insufficient review, outdated context, and challenges in interpreting and assigning the generated output.

Theme 4: Workflow Support and Perceived Usability

Participants raised multiple aspects related to how the artefact was perceived to support the decomposition workflow. As participants were guided through the workflow rather than operating the interface independently, these findings concern perceived usability.

Several suggestions concerned output presentation. P1 proposed shorter and more overviewable outputs, with the possibility of switching between a quick mode and a more detailed mode depending on the complexity of the requirement. P14 similarly emphasised the need for more concise outputs, noting that most artefact outputs contained more text than they would have desired. They highlighted the importance of keeping requirement descriptions efficient to read and interpret. P9 also proposed adding a simple flowchart or progress indicator showing the number of steps and the user's current position in the process.

In contrast, some participants raised positive points about the current workflow, for example P12 who perceived the tool as intuitive and easy to use during the walkthrough.

These suggestions reflect a focus on improving clarity and efficiency in how users understand, navigate, and interpret the generated outputs during decomposition.

Theme 5: Context Awareness and Decomposition Targeting

Participants suggested improvements related to how the artefact identifies, interprets, and uses contextual information during decomposition.

The most concrete improvement concerned receiver or PSE identification. P9 and P10 both argued that the tool should help determine where generated sub-requirements should be sent. P9 described the lack of PSE identification as a major blocker, while P10 suggested that the artefact could use existing requirement data to recommend potential receivers.

P14 raised a related concern regarding system context and decomposition positioning. They described uncertainty about where in the system hierarchy the decomposition was performed and how the generated sub-requirements relate to the surrounding system architecture. P12 expressed a similar concern, emphasising limitations in distinguishing between system scopes, abstraction levels, and applicability of retrieved requirements.

Several participants also emphasised the use of requirement attributes and metadata. P9 argued that requirements that are not safety- or legally relevant should not trigger unnecessary standards retrieval. P6 raised a related concern from the generation perspective, questioning whether requirements with different Automotive Safety Integrity Level classifications should result in different decompositions.

Scalability was also raised as a concern. P9 connected metadata use to handling

larger volumes of requirements. P14 similarly suggested that the tool should support working with collections of requirements, such as folders or requirement sets, rather than only individual inputs.

Finally, P9 proposed that the tool should identify weak or poorly formulated input requirements before decomposition begins, allowing them to be revised prior to further decomposition.

Theme 6: Trust in Correction-Based Learning

Participants were asked whether they would trust a system that learns from user corrections to support future requirements decomposition tasks. The responses were mixed.

Several participants expressed strong support for this functionality, including P9, P6, P13, and P12. P9 described learning from corrections as a way to improve consistency across engineers, particularly where decomposition practices vary. P6 similarly supported the idea of continuous learning from user input, while emphasising the need to maintain alignment with an underlying ground truth. P13 also viewed this as a key advantage of internal tools, where organisational data can be used to improve model behaviour.

Other participants expressed more cautious or conditional trust. P10 agreed with the concept but not strongly. P5 stated that such functionality would be acceptable if manual review remains part of the process. P15 described using such a system to obtain suggestions, but not fully trusting the output. P1 was the most cautious, particularly regarding the reuse of learned corrections across users.

Overall, participants described both potential benefits and limitations of correction-based learning, with some emphasising improved consistency and others expressing the need for manual review and careful use of learned knowledge.

8

Discussion

This chapter interprets the evaluation results in relation to the research questions, the artefact design, and threats to validity.

8.1 Answering RQ1: Combined Results from Interviews and Evaluation

The first research question concerned the perceived benefits and limitations of using LLM-based assistants to support requirements decomposition in automotive software development. As presented in Chapter 5, this question was primarily addressed through the interview study conducted during the relevance cycle. The purpose of revisiting RQ1 in this chapter is therefore not to re-answer the question independently, but to examine how the findings from the final evaluation align with, refine, or complicate the interview-based results.

The interview results indicated that the main expected benefit of an LLM-based decomposition assistant was not full automation of RE work, but support for cognitively demanding and repetitive parts of the decomposition process. The final evaluation aligns with this view, as participants generally assessed the artefact positively with respect to decomposition support and potential future use. Qualitative responses further framed the artefact as a reasoning aid or supporting partner, where generated outputs were compared against the engineer's own interpretation rather than accepted as final results. Several participants described value in using the workflow to explore alternative decomposition perspectives, support review activities, and surface contextual information relevant to the decomposition task. This aligns with prior RE research suggesting that LLM-based support is most useful when positioned as an expert-support mechanism rather than an autonomous replacement for engineering judgement [13, 55, 65].

The final evaluation also reinforced the importance of contextual grounding, which was a central limitation identified during the interviews. In the relevance cycle, participants emphasised that decomposition depends on system knowledge, standards, historical architectural decisions, and an understanding of the intended abstraction level. The evaluation results suggest that the artefact addressed parts of this need, particularly through the PSE Summary Agent, which received a positive assess-

ment. The RAG Agent was also assessed positively overall, although with more variation. This shows that participants valued grounding the decomposition workflow in system context and standards-related information. This is consistent with prior work showing that grounding mechanisms such as retrieval augmentation and structured contextualisation can improve the relevance of LLM-generated engineering artefacts [30, 59]. However, this benefit was not unconditional. Some qualitative responses highlighted that the usefulness of integrated retrieval depends partly on how efficiently contextual information can be incorporated into the workflow. These participants raised concerns regarding excessive steps, verbosity, and the amount of retrieved information, particularly for simpler decomposition tasks.

Even when contextual information is available, the final evaluation indicates that its usefulness depends on relevance and interpretability. This was most visible in the KG-RAG results, where responses were mixed regarding the relevance of historical decompositions and the amount of context provided for interpreting them. This refines the interview-based limitation regarding lack of domain context: retrieving semantically similar historical artefacts is not sufficient by itself. For historical reuse to support decomposition work, engineers also need contextual information that explains why a previous decomposition is relevant, under which system assumptions it was created, and how it relates to the current requirement.

The final evaluation was also consistent with the interview finding that abstraction-level judgement remains difficult. Participants generally assessed the decompositions positively, but abstraction-level fit showed weaker consensus than several other quality dimensions. This supports the interpretation that LLM assistance can help structure decomposition work, while expert judgement remains necessary for determining the appropriate level of detail. This also reflects findings from decomposition-oriented research, where abstraction-level transitions and hierarchy interpretation remain difficult even when structured prompting or decomposition guidance is provided [57, 58]. Qualitative responses reinforce this point by highlighting overreliance risks and the importance of maintaining critical evaluation of generated outputs, particularly in production-oriented contexts.

Taken together, the final evaluation supports the interview-based answer to RQ1: LLM-based assistants appear valuable when used as human-in-the-loop support for drafting, structuring, and reviewing decompositions. Their usefulness remains conditioned by the availability, relevance, and interpretability of contextual information, the difficulty of abstraction-level judgement, and the need for expert validation.

8.2 Interpreting Output Quality Through Requirements Quality Attributes

Table 7.2 connects the questionnaire items to requirements quality attributes adapted from Denger and Olsson [10], including correctness, consistency, completeness, right level of detail, comprehensibility and alignment with external constraints.

Correctness and consistency. Correctness and consistency were among the strongest quality dimensions. In Phase 1, all participants assessed the benchmark decomposition as logically consistent with the input requirement, while Phase 2 also showed a generally positive assessment, though with less complete agreement. This difference may be explained by the contextual nature of Phase 2, where participants judged the output against domain knowledge, system constraints, and technical expectations. The results suggest that the artefact often produced decompositions that were perceived as structurally coherent and consistent with the parent requirement. However, perceived consistency should not be interpreted as proof of domain correctness.

Completeness. Completeness was assessed positively overall, with participants generally considering the generated sub-requirements sufficient for fulfilling the input requirement. However, completeness was evaluated through expert judgement rather than comparison with a verified reference decomposition. The results should therefore be interpreted as evidence that the decompositions appeared sufficiently complete to participants during evaluation, not as proof that they were exhaustively complete.

Right level of detail. Right level of detail was one of the more uncertain quality dimensions. Although participants generally assessed the generated sub-requirements positively, abstraction-level fit showed weaker consensus than several other dimensions. This aligns with the interview findings, where level ambiguity and inconsistent detail were identified as central challenges in manual decomposition.
This result highlights that abstraction-level appropriateness is not purely linguistic. It depends on architectural knowledge, system boundaries, ownership, and the intended downstream use of the requirement. The artefact can support this judgement through hierarchy guidance and structured decomposition, but the evaluation does not show that it can determine the correct level of detail without domain-specific interpretation.

Comprehensibility. Comprehensibility was reflected in the evaluation of KDPs, Observations, and contextual information from the retrieval components. These elements were generally perceived as useful for understanding and reviewing the proposed decomposition. However, some participants noted that excessive verbosity in generated outputs can reduce clarity and make requirements harder to interpret, even when technically correct.

The KG-RAG results show that retrieved context is not automatically comprehensible. Historical decompositions require sufficient surrounding information for engineers to judge how they relate to the current requirement and whether they are reusable.

Alignment with external constraints. Participants tended to assess the outputs as technically valid and aligned with relevant safety or technical standards, particularly in relation to the RAG Agent's role in surfacing standards-related principles. However, responses were not uniformly positive, and the evaluation does not consti-

tute formal compliance verification. The results support standards-oriented retrieval as a useful design direction, while final responsibility for compliance remains with domain experts and established engineering processes.

Taken together, the artefact was perceived positively with respect to correctness, consistency, functional relevance, and explanatory support. The more uncertain dimensions were right level of detail and contextual comprehensibility for historical decomposition reuse. These findings provide the basis for discussing RQ2: the artefact's design choices were perceived as supporting several important quality attributes, but their usefulness depended on contextual grounding, transparent retrieval, and expert review.

8.3 Answering RQ2: Design Choices and Expert-Perceived Decomposition Quality

RQ2 is answered based on expert evaluation of the integrated artefact. Since no ablation study was conducted, the findings do not isolate causal effects of individual components. Instead, they show how participants perceived different design mechanisms as supporting or limiting decomposition quality within the combined workflow.

The PSE Summary Agent

The positive evaluation of the PSE Summary Agent suggests that this design choice addressed part of an important limitation identified during the interviews. Requirements decomposition in the automotive domain is difficult to perform from the input requirement alone, because interpretation often depends on system boundaries, architectural knowledge, terminology, and existing functional constraints. By summarising relevant PSE information before decomposition, the artefact provided additional context that participants could use to assess whether the generated requirements were meaningful for their system. This is consistent with qualitative feedback emphasising the importance of understanding system context and decomposition positioning when interpreting requirements.

This design choice was perceived as supporting the conditions under which correctness, completeness, and right level of detail could be assessed. However, its usefulness depends on the summary being accurate and relevant. The PSE Summary Agent should therefore be understood as a contextual support mechanism, not as a guarantee of correct decomposition.

Several qualitative responses emphasised the importance of identifying the intended decomposition target and surrounding system scope during decomposition. This suggests that contextual grounding is important not only for interpreting requirements, but also for determining how generated sub-requirements relate to the surrounding system hierarchy.

The RAG Agent

The RAG Agent was intended to ground the decomposition process in relevant standards and principles. Participants generally perceived this as useful, particularly for surfacing standards-related considerations relevant to the decomposition task. This suggests that RAG can support decomposition quality by introducing external constraints into the workflow, especially in a regulated automotive context.

However, the results were not unanimously positive for relevance. This can either indicate that retrieving standards-related information is not sufficient by itself or that the RAG corpus did not include the standard in the first place. The retrieved principles still need to be interpreted and connected to the specific decomposition task. RAG should therefore be treated as a mechanism for surfacing potentially relevant constraints, not as evidence of compliance.

Several participants also suggested that retrieval mechanisms should adapt more explicitly to requirement context and metadata, for example by considering safety relevance, legal applicability, or decomposition scope when selecting supporting information.

The KG-RAG Agent

The evaluation suggests that the KG-RAG Agent showed some ability to surface historical decompositions, but that participants were less convinced about the relevance and interpretability of the retrieved examples. The surrounding context was not consistently considered sufficient for evaluating how those historical decompositions should be applied.
This distinction is important for RQ2. Historical decomposition reuse may support quality, but only when retrieved examples include enough surrounding context. A previous decomposition is not self-explanatory; its relevance depends on system version, architectural assumptions, ownership boundaries, and rationale. This aligns with qualitative feedback indicating that historical decompositions are difficult to interpret without additional context regarding system scope, assumptions, and intended application.

The KG-RAG design choice therefore appears promising, but incomplete in its current form. Future versions should not only retrieve similar historical decompositions, but also inform why they are similar, what assumptions they depend on, and which parts are reusable for the current requirement.

Hierarchy Rules and Structured Prompt Orchestration

The positive results for logical consistency, functional relevance, and completeness are consistent with the intended role of hierarchy rules and structured prompt orchestration: to constrain the generated output and make the decomposition more inspectable. However, because the artefact was evaluated as an integrated workflow, these results cannot be attributed to these mechanisms in isolation.

The weaker consensus on abstraction-level fit shows the limit of this type of structural control. The results additionally suggest that structural guidance alone is insufficient when the intended decomposition target or surrounding system scope remains ambiguous. Hierarchy rules may reduce obvious level mismatches, but they cannot fully encode organisational conventions, architectural boundaries or knowledge. This further supports the interpretation that requirements decomposition cannot be treated as a purely deterministic transformation process, but instead depends on contextual engineering interpretation [63, 64].

Explanatory Output and Human Validation

KDPs, Observations, and Additional Information were included to make the generated decomposition more inspectable. Their positive evaluation suggests that participants valued explanatory support alongside the generated sub-requirements. Additionally, some participants described these explanatory elements as supporting comparison and reflection, reinforcing the role of the artefact as a reasoning aid rather than an authoritative source.
In this sense, human validation is supported through artefact transparency rather than added only as a separate final step. The tool provides structured candidate outputs, while the engineer remains responsible for judging correctness, relevance, and acceptance. Qualitative responses also highlighted concerns regarding over-reliance and reduced critical review if generated decompositions are accepted too readily, reinforcing the importance of inspectability and human validation within the workflow.

Summary of RQ2 Findings

Overall, the evaluation suggests that the artefact's design choices were perceived as supporting decomposition quality through contextual grounding, structural control, and inspectability. PSE summarisation and RAG supported system- and standards-related grounding, while hierarchy rules and structured orchestration were consistent with the goal of constraining the generated decomposition. KG-RAG showed potential for historical reuse, but also revealed that retrieved examples require stronger contextual explanation.

Since the evaluation assessed the integrated artefact, these findings should not be interpreted as causal evidence for individual components. Rather, they indicate that the combined workflow was perceived as useful for supporting expert-led requirements decomposition. Overall, the findings reinforce broader RE literature suggesting that contextual grounding, inspectability, and human validation are central design requirements for industrial LLM-assisted engineering workflows.

8.4 Threats to Validity

This section discusses threats to validity and the measures taken to mitigate them. The threats are categorised into reliability, construct validity, internal validity, ex-

ternal validity, and conclusion validity.

Reliability

Reliability concerns whether the study could be repeated with comparable results. A central threat is the non-deterministic nature of LLM-based systems, where the same input may produce different outputs. This was mitigated through structured prompts, predefined workflow steps, and fixed technical parameters where possible, although output variation cannot be fully eliminated.
Another reliability threat concerns the qualitative interview analysis. Since thematic coding involves interpretation, both authors reviewed and synthesised the interview findings. This helped maintain consistency and traceability between interview data, DOs, and evaluation criteria.

Construct Validity

Construct validity concerns whether the study measures the intended concepts. In this study, requirement quality was assessed through expert judgement rather than a verified ground-truth decomposition. As a result, the evaluation measures perceived correctness, completeness, consistency, and usefulness, not formally proven requirement quality.
To mitigate this, the questionnaire was mapped to requirements quality attributes adapted from Denger and Olsson [10], as shown in Table 7.2. However, the results should still be interpreted as expert-perceived quality rather than objective verification.
The same limitation applies to productivity and adoption. Participants evaluated the tool's perceived usefulness and future adoption potential, but the study did not measure long-term productivity effects in an operational setting.

Internal Validity

Internal validity concerns whether the findings can be attributed to the studied artefact. A key limitation is that the final evaluation assessed the integrated artefact as a whole, without conducting an ablation study to isolate the individual effects of PSE summarisation, RAG, KG-RAG, hierarchy rules, or structured prompt orchestration, or how these components interact.

Furthermore, the observed behaviour of the artefact results from the interaction between the underlying language model and the surrounding workflow design, including retrieval, contextual grounding, and prompt orchestration. This makes it difficult to disentangle whether improvements in perceived output quality are due to specific design mechanisms, the provided context, or the capabilities of the LLM itself.

Therefore, the study cannot make strong causal claims about individual components. Instead, the results indicate that the combined workflow was perceived as useful and that its outputs were generally assessed positively by domain experts.

Researcher involvement is another threat. The authors designed the artefact, conducted the evaluation, and interpreted the results. This was mitigated through a structured evaluation protocol, predefined questionnaire items, and explicit mapping between evaluation questions and quality attributes. Participants also did not operate the prototype independently; the researcher executed the workflow during the sessions. This ensured consistent execution, but limits conclusions about everyday usability and independent tool use.

Finally, another internal validity consideration concerns variation in evaluation difficulty between participants. During Phase 2, participants selected their own requirements for decomposition evaluation. Some intentionally selected particularly difficult or atypical requirements in order to stress-test the artefact under challenging conditions. In one case, the selected requirement depended heavily on external legal context that was not explicitly present in the requirement itself. In another case, a participant later considered the selected requirement too difficult to evaluate confidently within the scope of the session, and the corresponding Phase 2 and Phase 3 responses were therefore treated as neutral. These differences in requirement complexity likely contributed to variation in participant assessments and should be considered when interpreting the evaluation results.

External Validity

External validity concerns whether the findings transfer to other contexts. The study was conducted exclusively at Volvo Cars, using company-specific requirements, terminology, hierarchy rules, and domain knowledge. The empirical findings are therefore context-dependent and cannot be assumed to generalise directly to other organisations or domains.
However, the artefact architecture is not entirely Volvo-specific. Design mechanisms such as standards-oriented RAG, KG-RAG for historical reuse, structured orchestration, and HITL validation are conceptually transferable. What must be adapted is the local requirement hierarchy, terminology, source documents, and decomposition rules.
The study therefore supports analytical transferability rather than statistical generalisability. Similar workflows may be applicable in other engineering contexts, but further studies are needed to evaluate whether the observed benefits and limitations hold outside the Volvo Cars case.

8.5 Ethical Considerations

Interviews

Ethical considerations were addressed by following the principles of informed consent, confidentiality, beneficence, and other recommendations presented by Strandberg [75]. The following steps were taken:

- **Informed Consent:** Participants were informed through a consent form sent

out with the confirmation of the booked interview slot. The form clearly stated the purpose, the voluntary nature of participation, the right to withdraw, and how the data will be used. Consent was taken at the time of the interview and recorded.

- **Confidentiality:** Participants were assured that names and any identifying information would be removed from the transcripts. Only relevant domain experience in year ranges were kept. Any sort of publications would only use aggregated findings, anonymised quotations or pseudonyms.
- **Data handling:** Recordings and transcripts were stored on the company server with only the two interviewers and participant having access. Upon completion of the transcript, it was validated by the participant. Upon validation, the original recording was deleted, transcripts were retained and anonymised for research in agreement with the obtained consent.
- **Researcher skill:** Both researchers practised interview techniques through past experience and the pilot study to ensure competence.

Generative AI

The use of LLMs in the artefact introduced ethical considerations related to data governance and responsible use. The development was conducted using a company-approved enterprise LLM service within the approved research setting. When internal requirements, PSE information, or historical decomposition examples were used as LLM input, this was done within the approved setting.

A further ethical consideration concerns responsible use of LLM-generated outputs. Prior work on LLM-assisted software development suggests that interactions with LLMs can introduce or amplify cognitive biases, including preference for automated suggestions and reduced scrutiny of generated outputs [76]. In this study, this risk is relevant because generated decompositions may appear coherent and structured even when they are incomplete, outdated, or incorrectly scoped. To reduce this risk, the artefact presents outputs as candidate decompositions together with contextual and explanatory information intended to support critical inspection.

Computational sustainability was also considered during the artefact design. LLM inference has measurable computational and energy costs, particularly as models are repeatedly invoked in practical workflows [77]. The workflow limits the amount of context passed to the final generation step. This was intended to reduce unnecessary token usage and computational cost.

Use of generative AI

LLM-based generative AI tools were used in two contexts within this study. First, large language models were part of the artefact itself, where they were used to generate candidate requirements decompositions and related outputs. As a result,

the artefact's behaviour depends on the underlying model, which is non-deterministic and sensitive to prompt structure, model version, and parameter settings.

Second, AI-assisted tools were used during implementation and thesis writing, mainly for debugging, code suggestions, and grammar improvements. Since such tools may produce incorrect or misleading outputs, their use was limited to support purposes. All thesis content was written and reviewed by the authors, and AI-generated suggestions were carefully checked before acceptance.

The use of these tools may still have influenced implementation decisions and wording, which should be considered when interpreting the reproducibility of the study. In addition, the use of generative AI in safety-critical contexts introduces a risk that plausible but incorrect outputs may be accepted without sufficient scrutiny. This reinforces the importance of maintaining human oversight and validation in the workflow.

9

Conclusion

Requirements decomposition in automotive software engineering is a complex and context-dependent activity that requires architectural understanding, abstraction-level judgement, and domain expertise. The findings of this thesis suggest that LLM-based assistants can provide meaningful support for requirements decomposition when positioned as expert-assist tools rather than autonomous generation systems. The study indicates that decomposition support depends less on fluent requirement generation alone and more on whether generated outputs are grounded in relevant context, structured for review, and kept within a human-in-the-loop workflow.

With respect to RQ1, the study found that LLM-based assistants can support requirements decomposition by reducing cognitive effort, supporting initial drafting, retrieving contextual information, and providing alternative decomposition perspectives during refinement. Participants valued the artefact as a way to generate candidate decompositions that could be inspected, compared, and refined. The findings also suggest that such support can reduce context-switching and improve access to relevant engineering knowledge during decomposition activities. At the same time, the results showed that decomposition support remained highly sensitive to the quality, relevance, and interpretation of the available context. The most important limitations concerned abstraction-level judgement, incomplete contextual understanding, decomposition rationale verbosity, overreliance risks, and the continued need for engineers to critically review generated outputs before reuse in safety-critical engineering contexts.

Regarding RQ2, the findings further showed that design choices related to contextual grounding, hierarchy guidance, structured prompt orchestration, and explanatory outputs positively influenced expert-perceived decomposition quality. These mechanisms improved contextual alignment, reviewability, and decomposition consistency. The findings additionally suggest that useful decomposition support depends not only on textual understanding, but also on awareness of system hierarchy, subsystem responsibilities, and surrounding engineering context. Overall, contextual grounding and orchestration appeared more important to perceived usefulness than standalone language generation capability.

Participants generally perceived the artefact as most useful as a reasoning aid or second opinion rather than an autonomous decomposition system. The study also

indicates that requirements decomposition cannot be treated as a purely deterministic transformation problem. Decomposition practices and expectations varied across engineers and contexts. Participants described decomposition as partially subjective and dependent on engineering experience, subsystem perspective, and workflow context.
Expectations regarding workflow behaviour and output presentation also differed between participants. Some preferred more detailed contextual support, while others emphasised concise and efficient interaction. These findings reinforce the importance of HITL workflows where generated outputs remain reviewable, controllable, and subject to expert validation. LLM-based assistants should therefore be understood as decision-support tools for expert-led decomposition rather than sources of accepted requirements.
Although the proposed artefact improved contextual support and decomposition consistency in several areas, abstraction-level management remained a persistent challenge. In some cases, generated requirements were still perceived as either too abstract or too detailed relative to the intended target level. The findings additionally suggest that industrial usefulness depends not only on decomposition quality, but also on workflow integration, usability, and how contextual support is incorporated into existing engineering practices.

Future work could investigate more context-aware decomposition workflows with stronger understanding of system hierarchy, subsystem responsibilities, and surrounding engineering context. This includes retrieval and orchestration mechanisms that adapt contextual grounding based on requirement metadata, safety relevance, legal constraints, or decomposition target. Additional research could explore learning mechanisms based on user corrections and organisational decomposition practices, as well as improved abstraction-level reasoning and contextual targeting during decomposition.
Another relevant direction concerns broader integration into downstream engineering activities. Several participants highlighted the importance of architectural context and traceability throughout the development process. This highlights opportunities for workflows that support generation of additional downstream artefacts such as verification activities, test cases, or implementation-related engineering documentation. Longitudinal industrial deployment studies could additionally help improve understanding of how such tools are adopted, trusted, and incorporated into real-world requirements engineering workflows over time.

Bibliography

[1] Miroslaw Staron. *Automotive Software Development*, pages 67–95. Springer International Publishing, Cham, 2021.

[2] Grischa Liebel, Matthias Tichy, Eric Knauss, Oscar Ljungkrantz, and Gerald Stieglbauer. Organisation and communication problems in automotive requirements engineering. *Requirements Engineering*, 23(1):145–167, 2018.

[3] Matthias Weber and Joachim Weisbrod. Requirements Engineering in Automotive Development — Experiences and Challenges. *IEEE Software - SOFTWARE*, 20:331–340, 2002.

[4] Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. Large Language Models for Software Engineering: A Systematic Literature Review. *ACM Transactions on Software Engineering and Methodology*, 33(8), 2024.

[5] Alan Hevner and Samir Chatterjee. *Design Research in Information Systems: Theory and Practice*, volume 22. Springer Publishing Company, Incorporated, 2010.

[6] Roel J. Wieringa. *Design Science Methodology for Information Systems and Software Engineering*. Springer Publishing Company, Incorporated, 2014.

[7] Bashar Nuseibeh and Steve Easterbrook. Requirements engineering: a roadmap. In *Proceedings of the Conference on The Future of Software Engineering*, ICSE '00, page 35–46, New York, NY, USA, 2000. Association for Computing Machinery.

[8] Ian Sommerville and Pete Sawyer. *Requirements Engineering: A Good Practice Guide*. John Wiley & Sons, Inc., USA, 1st edition, 1997.

[9] Karl E Wiegers and Joy Beatty. *Software Requirements 3*. Microsoft Press, 2013.

[10] Christian Denger and Thomas Olsson. Quality assurance in requirements engineering. In *Engineering and Managing Software Requirements*, pages 163–185. Springer, 2005.

[11] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of Hallucination in Natural Language Generation. *ACM Computing Surveys*, 55(12):1–38, 2023.

[12] Vladimir Geroimenko. *The Essential Guide to Prompt Engineering: Key Principles, Techniques, Challenges, and Security Risks*. SpringerBriefs in Computer Science. Springer Nature Switzerland, Cham, 1 edition, 2025.

[13] Juan C. Yelmo, Yod-Samuel Martín, and Santiago Perez-Acuna. Experimental Evaluation of AI-Augmented Cybersecurity Requirements Generation Leveraging LLMs' Capabilities. *IEEE Access*, 14:19579–19606, 2026.

[14] O. Gotel and A. Finkelstein. Extended requirements traceability: results of an industrial case study. In *Proceedings of ISRE '97: 3rd IEEE International Symposium on Requirements Engineering*, pages 169–178, 1997.

[15] S. Magnus Ågren, Eric Knauss, Rogardt Heldal, Patrizio Pelliccione, Gösta Malmqvist, and Jonas Bodén. The Manager Perspective on Requirements Impact on Automotive Systems Development Speed. In *2018 IEEE 26th International Requirements Engineering Conference (RE)*, pages 17–28, 2018.

[16] International Organization for Standardization. Road vehicles – Functional safety (ISO 26262:2018 series), 2018. Consists of Parts 1–12.

[17] International Organization for Standardization and SAE International. Road vehicles – Cybersecurity engineering (ISO/SAE 21434:2021), 2021.

[18] Jin LC Guo, Jan-Philipp Steghöfer, Andreas Vogelsang, and Jane Cleland-Huang. Natural Language Processing for Requirements Traceability. In *Handbook on Natural Language Processing for Requirements Engineering*, pages 89–116. Springer, 2025.

[19] Hina Saeeda, Zuzana Rohacova, Oskar Jakobsson, Hans-Martin Heyn, Eric Knauss, Alessia Knauss, and Jennifer Horkoff. Requirements Representations in Machine Learning-Based Automotive Perception Systems Development for Multi-party Collaboration. In Anne Hess and Angelo Susi, editors, *Requirements Engineering: Foundation for Software Quality*, pages 197–213, Cham, 2025. Springer Nature Switzerland.

[20] David P. Kirkman. Requirement decomposition and traceability. *Requirements Engineering*, 3(2):107–114, 1998.

[21] Robert F. Bordley. *Requirement Decomposition*, pages 79–90. Springer Nature Switzerland, Cham, 2026.

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *arXiv preprint arXiv:1706.03762*, 2023.

[23] Tong Xiao and Jingbo Zhu. Foundations of large language models. *arXiv preprint arXiv:2501.09223*, 2025.

[24] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint arXiv:2302.13971*, 2023.

[25] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A Survey of Large Language Models. *Frontiers of Computer Science*, 20(12):2012627, 2026.

[26] Michael Townsen Hicks, James Humphries, and Joe Slater. Chatgpt is bullshit. *Ethics and Information Technology*, 26(2), 2024.

[27] Isabelle Augenstein, Timothy Baldwin, Meeyoung Cha, Tanmoy Chakraborty, Giovanni Luca Ciampaglia, David Corney, Renee DiResta, Emilio Ferrara, Scott Hale, Alon Halevy, Eduard Hovy, Heng Ji, Filippo Menczer, Ruben Miguez, Preslav Nakov, Dietram Scheufele, Shivam Sharma, and Giovanni Zagni. Factuality challenges in the era of large language models and opportunities for fact-checking. *Nature Machine Intelligence*, 6(8):852–863, 2024.

[28] Ggaliwango Marvin, Nakayiza Hellen, Daudi Jjingo, and Joyce Nakatumba-Nabende. Prompt engineering in large language models. In I. Jeena Jacob, Selwyn Piramuthu, and Przemyslaw Falkowski-Gilski, editors, *Data Intelligence and Cognitive Informatics*, pages 387–402, Singapore, 2024. Springer Nature Singapore.

[29] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Computing Surveys*, 55(9), 2023.

[30] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[31] Jun Zhao, Yongzhuo Yang, Xiang Hu, Jingqi Tong, Yi Lu, Wei Wu, Tao Gui, Qi Zhang, and Xuanjing Huang. Understanding Parametric and Contextual Knowledge Reconciliation within Large Language Models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[32] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, 2019. Association for Computational Linguistics.

[33] S Joshua Johnson, M Ramakrishna Murty, and I Navakanth. A detailed review on word embedding techniques with emphasis on word2vec. *Multimedia Tools and Applications*, 83(13):37979–38007, 2024.

[34] Markus Reuter, Tobias Lingenberg, Ruta Liepina, Francesca Lagioia, Marco Lippi, Giovanni Sartor, Andrea Passerini, and Burcu Sayin. Towards Reliable Retrieval in RAG Systems for Large Legal Datasets. In Nikolaos Aletras, Ilias Chalkidis, Leslie Barrett, Cătălina Goanță, Daniel Preoțiuc-Pietro, and Gerasimos Spanakis, editors, *Proceedings of the Natural Legal Language Processing Workshop 2025*, pages 17–30, Suzhou, China, 2025. Association for Computational Linguistics.

[35] Md Saleh Ibtasham, Sarmad Bashir, Muhammad Abbas, Zulqarnain Haider, Mehrdad Saadatmand, and Antonio Cicchetti. ReqRAG: Enhancing Software Release Management through Retrieval-Augmented LLMs: An Industrial Study. In Anne Hess and Angelo Susi, editors, *Requirements Engineering: Foundation for Software Quality*, pages 277–292, Cham, 2025. Springer Nature Switzerland.

[36] Scott Barnett, Stefanus Kurniawan, Srikanth Thudumu, Zach Brannelly, and Mohamed Abdelrazek. Seven Failure Points When Engineering a Retrieval Augmented Generation System. In *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering - Software Engineering for AI*, CAIN '24, page 194–199, New York, NY, USA, 2024. Association for Computing Machinery.

[37] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia D'amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. Knowledge Graphs. *ACM Computing Surveys*, 54(4):1–37, 2021.

[38] Miranda Rodríguez, Ali Nouri, Zhennan Fei, and Maria M. Hedblom. An LLM and Embeddings-Based Multi-agentic System for Knowledge Graph Construction and Verification. In *International Conference on Principles and Practice of Multi-Agent Systems*, pages 400–417. Springer, 2025.

[39] Yufan Dang, Chen Qian, Xueheng Luo, Jingru Fan, Zihao Xie, Ruijie Shi, Weize Chen, Cheng Yang, Xiaoyin Che, Ye Tian, Xuantang Xiong, Lei Han, Zhiyuan Liu, and Maosong Sun. Multi-Agent Collaboration via Evolving Orchestration. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi,

and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38, pages 165025–165059. Curran Associates, Inc., 2025.

[40] Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O'Sullivan, and Hoang D. Nguyen. Multi-agent collaboration mechanisms: A survey of llms. *arXiv preprint arXiv:2501.06322*, 2025.

[41] Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. AgentVerse: Facilitating Multi-Agent Collaboration and Exploring Emergent Behaviors. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun, editors, *International Conference on Learning Representations*, volume 2024, pages 20094–20136, 2024.

[42] Qing Ye and Jing Tan. Agent Contracts: A Formal Framework for Resource-Bounded Autonomous AI Systems. *arXiv preprint arXiv:2601.08815*, 2026.

[43] Iman Abbasnejad, Xuefeng Liu, and Atanu Roy. Deciding the Path: Leveraging Multi-Agent Systems for Solving Complex Tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 4255–4264, 2025.

[44] Andrea Wohlgemuth, Fabiano Dalpiaz, and Erik Kamsties. Exploring and Characterizing Ad-Hoc Requirements - A Case Study at a Large-Scale Systems Provider. In Anne Hess and Angelo Susi, editors, *Requirements Engineering: Foundation for Software Quality*, pages 72–88, Cham, 2025. Springer Nature Switzerland.

[45] Minghui Sun, Georgios Bakirtzis, Hassan Jafarzadeh, and Cody Fleming. Correct-by-construction requirement decomposition. *Software and Systems Modeling*, 25(1):271–286, 2026.

[46] Khan Mohammad Habibullah, Hans-Martin Heyn, Gregory Gay, Jennifer Horkoff, Eric Knauss, Markus Borg, Alessia Knauss, Håkan Sivencrona, and Polly Jing Li. Requirements and software engineering for automotive perception systems: an interview study. *Requirements Engineering*, 29(1):25–48, 2024.

[47] Md. Asraful Haque. LLMs: A game-changer for software engineers? *BenchCouncil Transactions on Benchmarks, Standards and Evaluations*, 5(1):100204, 2025.

[48] Shuaicai Ren, Hiroyuki Nakagawa, and Tatsuhiro Tsuchiya. Combining Prompts with Examples to Enhance LLM-Based Requirement Elicitation. In *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 1376–1381, 2024.

[49] Camila Almeida, Isaque Copque, Alvaro Oliveira, Murilo Arouca, Adriano Bar-

bosa, Sávio Freire, Manoel Mendonça, and Julio Leite. From Elicitation Interviews to Software Requirements: Evaluating LLM Performance in Requirement Generation. In *Workshop on Requirements Engineering (WER)*, 2025.

[50] Satoshi Masuda, Satoshi Kouzawa, Kyousuke Sezai, Hidetoshi Suhara, Yasuaki Hiruta, and Kunihiro Kudou. Generating High-Level Test Cases from Requirements Using LLM: an Industry Study. In *2026 International Conference on Artificial Intelligence, Computer, Data Sciences and Applications (ACDSA)*, pages 1–9, 2026.

[51] Vibhashree Hippargi, Erik Kamsties, and Jürgen Naumann. Evaluating the Capabilities of LLMs in Traceability Maintenance for Automotive System and Software Requirements: Three Case Studies. In *Softwaretechnik-Trends Band 45, Heft 1*. Gesellschaft für Informatik e.V., 2025.

[52] Ali Nouri, Beatriz Cabrero-Daniel, Zhennan Fei, Krishna Ronanki, Håkan Sivencrona, and Christian Berger. Large Language Models in Code Co-generation for Safe Autonomous Vehicles. In *International Conference on Computer Safety, Reliability, and Security*, pages 193–208. Springer, 2025.

[53] Alcino Cunha and Nuno Macedo. Validating Formal Specifications with LLM-generated Test Cases. In *International Symposium on Formal Methods*, pages 279–297. Springer, 2026.

[54] Yilmaz Uygun and Victor Momodu. Local large language models to simplify requirement engineering documents in the automotive industry. *Production & Manufacturing Research*, 12, 2024.

[55] Orhan Sirin, Malik Abdul Sami, Tuomas Granlund, Jussi Rasku, Zheying Zhang, and Pekka Abrahamsson. Enhancing Regulation-Adherent Requirement Engineering with Contextual AI: An Industrial Study. In Giuseppe Scanniello, Valentina Lenarduzzi, Simone Romano, Sira Vegas, and Rita Francese, editors, *Product-Focused Software Process Improvement. Industry, Doctoral-Symposium, Tutorial, and Workshop Papers*, pages 69–85, Cham, 2026. Springer Nature Switzerland.

[56] Chetan Arora, Fanyu Wang, Chakkrit Tantithamthavorn, Aldeida Aleti, and Shaun Kenyon. From Domain Documents to Requirements: Retrieval-Augmented Generation in the Space Industry. In *2025 IEEE 33rd International Requirements Engineering Conference (RE)*, pages 303–307. IEEE, 2025.

[57] Anamaria-Roberta Preda, Christoph Mayr-Dorn, Atif Mashkoor, and Alexander Egyed. Supporting high-level to low-level requirements coverage reviewing with large language models. In *Proceedings of the 21st International Conference on Mining Software Repositories*, MSR '24, page 242–253, New York, NY, USA, 2024. Association for Computing Machinery.

[58] Cheol Young Park, Shou Matsumoto, Hong Seok Park, Youngjoon Oh, and

Joongyoon Lee. When to stop decomposing: Llm-assisted quality gates for functional decomposition in systems engineering. *IEEE Access*, 14:57427–57443, 2026.

[59] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From Local to Global: A Graph RAG Approach to Query-Focused Summarization. *arXiv preprint arXiv:2404.16130*, 2025.

[60] Lukas Bahr, Christoph Wehner, Judith Wewerka, José Bittencourt, Ute Schmid, and Rüdiger Daub. Knowledge graph enhanced retrieval-augmented generation for failure mode and effects analysis. *Journal of Industrial Information Integration*, 45:100807, 2025.

[61] Diego Sanmartin. KG-RAG: Bridging the Gap Between Knowledge and Creativity. *arXiv preprint arXiv:2405.12035*, 2024.

[62] Hong Qing Yu and Frank McQuade. RAG-KG-IL: A Multi-Agent Hybrid Framework for Reducing Hallucinations and Enhancing LLM Reasoning through RAG and Incremental Knowledge Graph Learning Integration. *arXiv preprint arXiv:2503.13514*, 2025.

[63] Malik Abdul Sami, Muhammad Waseem, Zheying Zhang, Zeeshan Rasheed, Kari Systä, and Pekka Abrahamsson. AI based multiagent approach for requirements elicitation and analysis. *arXiv preprint arXiv:2409.00038*, 2024.

[64] Marc Oriol, Quim Motger, Jordi Marco, and Xavier Franch. Multi-Agent Debate Strategies to Enhance Requirements Engineering with Large Language Models. In *2025 IEEE 33rd International Requirements Engineering Conference (RE)*, page 527–534. IEEE, 2025.

[65] Jiangping Huang, Dongmin Jin, Weisong Sun, Yang Liu, and Zhi Jin. Envisioning Intelligent Requirements Engineering via Knowledge-Guided Multi-Agent Collaboration. In *Proceedings of the 40th International Conference on Automated Software Engineering-New Ideas and Emerging Results Track*, pages 1–6. IEEE Computer Society, 2025.

[66] Mohammad Amin Zadenoori, Liping Zhao, Waad Alhoshan, and Alessio Ferrari. Automatic Prompt Engineering: The Case of Requirements Classification. In Anne Hess and Angelo Susi, editors, *Requirements Engineering: Foundation for Software Quality*, pages 217–225, Cham, 2025. Springer Nature Switzerland.

[67] Claes Wohlin. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, EASE '14, New York, NY, USA, 2014. Association for Computing Machinery.

[68] Per Runeson and Martin Höst. Guidelines for conducting and reporting case

study research in software engineering. *Empirical Softw. Engg.*, 14(2):131–164, 2009.

[69] Ali Nouri, Christian Berger, and Fredrik Törner. On STPA for Distributed Development of Safe Autonomous Driving: An Interview Study. In *2023 49th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, page 5–12. IEEE, 2023.

[70] Cormac McGrath, Per J. Palmgren, and Matilda Liljedahl. Twelve tips for conducting qualitative research interviews. *Medical Teacher*, 41(9):1002–1006, 2019. PMID: 30261797.

[71] Juliet Corbin and Anselm Strauss. *Basics of qualitative research: Techniques and procedures for developing grounded theory.* Sage Publications, Los Angeles, CA, 3rd edition, 2008.

[72] Miroslaw Staron. Requirements Engineering for Automotive Embedded Systems. In *Automotive Systems and Software Engineering: State of the Art and Future Trends*, pages 11–28. Springer, 2019.

[73] Fred D. Davis. Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *Management Information Systems Quarterly*, 13(3):319–340, 1989.

[74] IEEE. IEEE Recommended Practice for Software Requirements Specifications, 1998. IEEE Std 830-1998.

[75] Per Erik Strandberg. Ethical interviews in software engineering. In *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 1–11. IEEE, 2019.

[76] Xinyi Zhou, Zeinadsadat Saghi, Sadra Sabouri, Rahul Pandita, Mollie McGuire, and Souti Chattopadhyay. Cognitive Biases in LLM-Assisted Software Development. *arXiv preprint arXiv:2601.08045*, 2026.

[77] Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadepally. From Words to Watts: Benchmarking the Energy Costs of Large Language Model Inference. In *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–9. IEEE, 2023.

A

Appendix 1: Prompt Engineering Principles and Techniques

This appendix summarises prompt engineering techniques considered during the design of the artefact. The additional details below document techniques that informed implementation-level prompt design. The techniques summarised in Table A.1 are based on established prompt engineering concepts such as structured instructions, examples, role prompting, constraints, and structured output specifications [12, 13, 28, 29].

Table A.1: Prompt Engineering Techniques Considered in the Artefact Design.

Technique	**Purpose**	**Use in the artefact**
Direct instructions and action verbs	Clarify the intended model behaviour.	Used to specify tasks such as summarising context, preserving intent, and generating candidate lower-level requirements.
Persona-based prompting	Frames the model response towards a specific professional role.	Used where appropriate to orient the model towards RE and automotive engineering terminology.
Separation of prompt components	Distinguishes instructions, user input, retrieved context, constraints, and output requirements.	Used in structured prompt orchestration to reduce ambiguity between contextual sources.
Few-shot prompting	Provides examples of the expected pattern or output style.	Helps the model infer the patterns from the examples for the intended output, which is especially useful for domain-specific tasks.
Instructional guidelines and negative constraints	Specify what the model should avoid.	Used to reduce unsupported assumptions, irrelevant retrieved context, and invented details.

Continued on next page

Table A.1: Prompt engineering techniques considered in the artefact design. Continued.

Technique	**Purpose**	**Use in the artefact**
Structured output specification	Constrains the response format.	Used to produce candidate requirements, key decision points, observations, and additional context in predefined fields.
Applicability filtering	Assesses whether retrieved information is relevant to the current task.	Used to reduce noise from retrieved standards or historical requirements before final generation.

B

Appendix 2: Interview Guide

This appendix presents the interview guide used to gather qualitative data from systems engineers in the automotive industry. The guide was designed to ensure consistency across interview rounds and focus on the practical challenges of requirements decomposition.

Methodology and Participant Information

- **Interview Method:** The interviews will be conducted and recorded through Microsoft Teams. The transcription will be automatically created and adjustments will be made to correct the automated transcript in case of errors by the interviewers. This corrected transcription will then be validated by the interviewee to ensure correctness.
- **Pilot Interview:**Before the interviews a pilot interview will be conducted to determine the performance of the interview guide and to reach maturity for the questions. The results for the pilot interview are not included for the final data analysis. This pilot is done to test the structure, efficacy, and relevance of the questions asked during the interview.
- **Participant Pool:** Participants are system engineers from the automotive industry who are performing requirements engineering.
- **Ethics and Communication:** Participants are informed of their right to withdraw consent at any time. Data is handled transparently, with an information sheet provided prior to the session.

Interview Structure

This section presents the order of the interview questions.

Welcoming and Ethics Confirmation

Before the main questions, a pre-recorded video is played to ensure consistency and avoid researcher bias. The following are confirmed on record:

- Explicit consent for recording and transcript validation.
- Awareness of data storage and deletion policies.

Demographics

1. Years of experience (YoE) in the company and domain?
2. YoE working specifically with requirements?
3. YoE working with Generative AI (genAI)?
4. Description of current role and professional background.

Main Interview subsections

This section presents the questions for the main part of the interview.

Part 1: General Requirements Decomposition

Definition: Requirements decomposition concerns the practice of creating multiple sub-requirements from a requirement, typically associated with increased technical granularity.

1. How are you doing requirements decomposition in your current role?
    - When going from e.g., a [higher level] to [lower level], what is your approach?
    - How much time do you spend on requirements decomposition?
2. What are the current challenges with requirements decomposition today?
    - Process itself? Collaboration with external partners?
3. Walk me through your first steps when you receive a new requirement. What do you do first?
    - Do you start from a blank state or look at similar requirements?
    - What do you do if you are missing domain knowledge?
4. Which standards or guidelines do you use when breaking down a requirement?
5. How do you determine when a requirement has been sufficiently broken down?

Part 2: Quality of Requirements

1. How do you ensure that new requirements are of adequate quality?

- How do you ensure consistency between requirements?
- What challenges do you face when defining unambiguous requirements?
  - How do you make sure they are interpreted correctly?

2. How do time and resource constraints impact the level of detail of your requirements?
   - What trade-offs do you make between speed and accuracy?

Part 3: Generative AI subsection

1. To what extent do you use generative AI in your daily workflow?
   - Which tools specifically?
   - Which tools specifically for requirements decomposition?
2. What do you do when you receive generated content that you are not satisfied with?
   - What steps do you take to enhance it (e.g., provide more context/material)?
3. How has genAI affected your productivity?
4. Assume there is an ideal AI assistant for requirements decomposition—what functionalities would you expect?
5. What else would you like to share about the process?

C

Appendix 3: Final Evaluation Questionnaire

This appendix contains the full questionnaire used during the final evaluation of the artefact.

Demographic Information

1. Years of experience in Volvo Cars and automotive software development
2. Years of experience working with requirements
3. Years of experience working with generative AI in your workflow
4. Description of current role and background

Phase 1: Benchmark Evaluation

Step 1: Input Requirement

1. Are you familiar with this requirement or the associated PSE?
    - Yes
    - No
2. Can you assess alignment with internal system design for decompositions of this requirement?
    - Yes
    - No

Step 2: Generated Decomposition

Participants were asked to rate the following statements on a 5-point Likert scale (Strongly disagree, Disagree, Neither agree nor disagree, Agree, Strongly agree):

- The sub-requirements are logically consistent with the input requirement
- The sub-requirements are at the correct level of abstraction
- The sub-requirements have distinct functional responsibilities
- The sub-requirements are sufficient to fulfil the input requirement
- The sub-requirements include functionalities not related to the input requirement
- The key decision points help to understand and validate the decomposition
- The observations and additional information provide useful context
- The decomposition is logically consistent with the input requirement

If applicable:

- The outputs are technically valid and compliant with relevant system design and standards

Phase 2: Contextual Evaluation

Step 1: Requirement Selection

1. Chosen requirement identifier
2. Chosen PSE identifier

Participants evaluated:

- The generated PSE summary is accurate
- The PSE summary captures relevant information for the decomposition

Step 2: RAG Agent Output

- The suggested principles are correct
- The suggested principles are relevant for the decomposition
- The RAG agent facilitates identification of relevant standards

Step 3: KG-RAG Agent Output

- The provided information is sufficient to understand the historical context
- The suggested historical decompositions are relevant
- The KG-RAG agent facilitates identification of relevant prior decompositions

Step 4: Final Decomposition

Participants were asked to rate:

- The decomposition is logically consistent with the input requirement
- The sub-requirements are at the correct level of abstraction
- The sub-requirements have distinct functional responsibilities
- The sub-requirements are sufficient to fulfil the input requirement
- The sub-requirements include functionalities not related to the input requirement
- The key decision points help to understand and validate the decomposition
- The observations and additional information provide useful context
- The outputs are technically valid and compliant with relevant standards

Participants also rated:

- I would intend to use this tool in future requirements decomposition tasks
- I would recommend this tool to colleagues
- Using this tool would improve the quality of my requirements decompositions
- The tool was intuitive to interact with
- Significant training would be required to use this tool effectively

Phase 3: Open-Ended Questions

1. How do you see this tool as a collaborative partner?
2. Does the reduced context-switching improve your workflow?
3. What is the biggest risk of using this tool in a safety-critical project?
4. What improvements would you suggest?

5. Would you trust a system that learns from your corrections?

6. Additional comments

D

Appendix 4: Implemented Technical Parameters

Table D.1 summarises selected implementation parameters and architectural mechanisms used in the final artefact.

Table D.1: Selected Implementation Parameters and Architectural Mechanisms in the Implemented Artefact

Component / Parameter	**Configuration**	**Role in Architecture**
RAG chunk context augmentation	Prepended document summaries	Injects document-level context into embeddings to improve retrieval relevance.
RAG retrieval depth	`top_k` = 5	Controls the number of retrieved standards and guideline chunks incorporated into contextual grounding.
RAG document store	PostgreSQL + pgvector	Stores embedded standards and guidelines chunks for semantic similarity retrieval.
KG seed retrieval depth	`retrieval_top_n` = 3	Controls the number of semantically similar historical requirements used as graph expansion seeds.
KG requirement store	PostgreSQL + pgvector	Stores embedded requirement descriptions for semantic similarity retrieval.
KG graph store	Neo4j	Supports traversal and reconstruction of historical decomposition relationships.
LLM generation setings	temperature = 0, `top_p` = 1, seed = 42	Supports more deterministic and reproducible schema-constrained generation.
Output schema	Structured schema	Enforces predictable decomposition structure and output formatting.