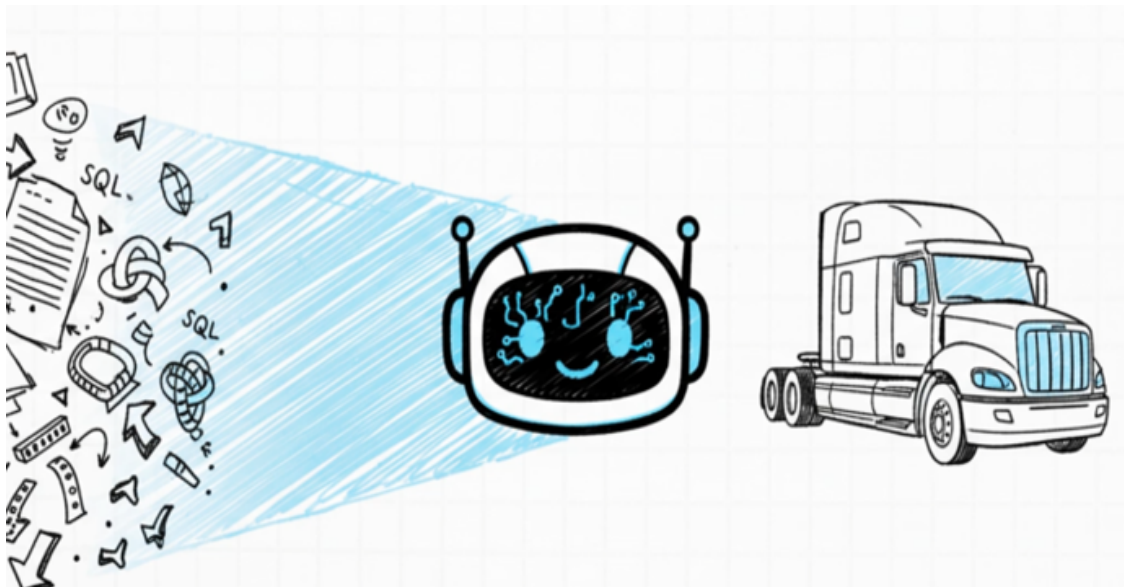




CHALMERS
UNIVERSITY OF TECHNOLOGY



Design and Empirical Evaluation of a Multi-Agent System for Engineering Workflows

A Case Study in Early-Phase Truck Integration

Master's thesis in: Complex Adaptive Systems and Mobility Engineering

Lingyue Zhang, Mengbing Qiu

DEPARTMENT OF MECHANICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY

Gothenburg, Sweden 2026

www.chalmers.se

MASTER'S THESIS 2026

Design and Empirical Evaluation of a Multi-Agent System for Engineering Workflows

A Case Study in Early-Phase Truck Integration

Lingyue Zhang
Mengbing Qiu



Department of Mechanical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026

Design and Empirical Evaluation of a Multi-Agent System for Engineering Workflows

A Case Study in Early-Phase Truck Integration

Lingyue Zhang, Mengbing Qiu

© Lingyue Zhang, Mengbing Qiu 2026.

Supervisor: Wenting Hu, Volvo Group Trucks Technology

Examiner: Kristina Wärmefjord, Professor, Chalmers University of Technology

Master's Thesis 2026

Department of Mechanical Engineering

Chalmers University of Technology

SE-412 96 Gothenburg

Telephone +46 31 772 1000

Typeset in L^AT_EX

Printed by Chalmers Reproservice

Gothenburg, Sweden 2026

Design and Empirical Evaluation of a Multi-Agent System for Engineering Workflows - A Case Study in Early-Phase Truck Integration

LINGYUE ZHANG, MENGBING QIU

Department of Mechanical Engineering

Chalmers University of Technology

Abstract

Early-phase truck integration requires engineers to coordinate vehicle data, regional regulations, engineering calculations, and CAD-related operations across heterogeneous tools. This thesis designs, implements, and empirically evaluates a tool-orchestrated multi-agent system for a bounded set of these tasks. The research examines heterogeneous-resource integration through a traceable and extensible tool-orchestrated architecture, correct workflow execution and contract conformance on the defined benchmark, and trade-offs in accuracy, evidence faithfulness, reliability, and latency across tasks and configurations.

The prototype uses a LangGraph supervisor-specialist architecture comprising one orchestrator and five ReAct subagents for Database, Regulation, Payload, Agility, and CAD tasks. A separate non-agent utility node manages the RAG knowledge base. The study follows an artifact-centred design-science strategy. Evaluation is conducted at the offline, component, system, and implementation levels using frozen datasets. These datasets include a simulated vehicle database, a curated local regulation corpus, deterministic reference calculations, and explicit workflow checks.

On the defined benchmark, routing accuracy was 91.0% across 100 requests, and 49 of 50 end-to-end tasks passed the required sequence and output checks. The Payload calculation results agreed with an independent oracle in all 60 evaluated cases. In the Database evaluation, the required values were retrieved more often than the generated result exactly matched the reference result. This difference demonstrates why capability-specific metrics are needed. At retrieval depth five, the annotated answer fact was found for 83.65% of regulation queries, whereas the specifically annotated source was retrieved for 64.42%. The paired automatic comparison between rule-only and RAG-only regulation answering did not detect a statistically significant difference in term match ($p = 0.1263$), while rule-only execution had a substantial latency advantage.

The results support the feasibility of bounded multi-agent orchestration for engineering assistance. They do not establish production performance or superiority over a monolithic agent. The quantitative findings are limited to the synthetic benchmark and small document corpus. The CAD and Agility measurements cover planning rather than complete geometry execution. The study therefore supports a capability-specific interpretation in which deterministic operations, retrieved evidence, execution traces, and continued human oversight remain necessary for dependable engineering use.

Keywords: Generative AI; Large Language Models; LLM Agents; Vehicle Integration; Engineering Automation; Multi-Agent System; Tool Orchestration; Retrieval-Augmented Generation

Acknowledgements

We would like to express our sincere gratitude to everyone who supported and contributed to this thesis project.

First, we would like to thank Volvo Group Trucks Technology for providing us with the opportunity to conduct this thesis in an industrial environment. We are especially grateful to Mikael Rydh and Wenting Hu for their guidance, support, and valuable feedback throughout the project. Their insights into vehicle architecture and engineering practice were important in shaping both the direction and practical relevance of this work.

We would also like to thank Professor Kristina Wärmefjord and Dr. Kostas Stylidis for their academic supervision, constructive feedback, and support throughout the thesis process. We are also grateful to the colleagues and contributors who shared their knowledge, participated in discussions, and helped us better understand the engineering context and technical challenges addressed in this study.

Finally, we would like to thank our families and friends for their encouragement, patience, and support throughout our studies and during the completion of this thesis.

Lingyue Zhang and Mengbing Qiu, Gothenburg, September 2026

List of Acronyms

The acronyms used in this thesis are listed below in alphabetical order.

AI	Artificial Intelligence
API	Application Programming Interface
BM25	Best Match 25 ranking function
CAD	Computer-Aided Design
COM	Component Object Model
EU	European Union
GPT	Generative Pre-trained Transformer
HTTP	Hypertext Transfer Protocol
ID	Identifier
KB	Knowledge Base
LLM	Large Language Model
MRR	Mean Reciprocal Rank
NDCG	Normalised Discounted Cumulative Gain
QA	Question Answering
RAG	Retrieval-Augmented Generation
ReAct	Reasoning and Acting
RQ	Research Question
SQL	Structured Query Language
UI	User Interface
URL	Uniform Resource Locator

Nomenclature

The following notation is used in this thesis. Acronyms are listed separately.

Evaluation notation

k	Retrieval depth; the number of top-ranked chunks considered for a query
N	Number of observations or questions in the evaluated sample
p	Probability value reported by a statistical hypothesis test
p_{95}	Empirical 95th percentile of the latency distribution

Retrieval and classification measures

FactHit@ k	Proportion of queries for which the first k retrieved chunks contain the annotated answer fact
Macro-F1	Unweighted mean of the class-specific F1 scores
MRR	Mean reciprocal rank of the first relevant retrieved source; a query with no relevant retrieved source contributes zero
NDCG@ k	Normalised discounted cumulative gain computed over the first k retrieved results
Precision@ k	Proportion of the first k retrieved results that match an annotated relevant source
Recall@ k	Proportion of annotated relevant sources found among the first k retrieved results

Contents

List of Acronyms	ix
Nomenclature	xi
List of Figures	xvii
List of Tables	xxi
1 Introduction	1
1.1 Background and Motivation	1
1.2 Problem Definition	2
1.3 Research Aim	3
1.4 Research Questions	3
1.5 Contributions	3
1.6 Scope and Delimitations	4
1.7 Thesis Structure	4
1.8 AI Disclaimer	4
2 Theory	5
2.1 Early-Phase Vehicle Integration and Requirements	5
2.2 Large Language Models and Agentic Workflows	5
2.3 Multi-Agent Systems and Orchestration	6
2.4 Natural-Language Access to Structured Engineering Data	6
2.5 Retrieval-Augmented Generation for Regulatory Knowledge	7
2.6 Engineering Automation and Human Oversight	7
2.7 Synthetic Data for Confidential Engineering Systems	8
2.8 Synthesis and Research Gap	8
3 Research Methodology	9
3.1 Research Design	9
3.1.1 Design-Science Case Study Strategy	9
3.1.2 Industrial Context and Unit of Analysis	10
3.2 Problem Investigation	10
3.2.1 Development Data and Task Selection	10
3.2.2 Representing Workflows for Agent Execution	11
3.3 Design Objectives	12
3.4 Artifact Development	12

3.4.1	Data and Confidentiality Strategy	13
3.5	System Demonstration and Evaluation	13
3.5.1	Evaluation Design Overview	13
3.5.2	Offline and Contract Validation	14
3.5.2.1	Levels of Verifiability	14
3.5.2.2	Contract Conformance as Capability Evidence	14
3.5.3	Component-Level Evaluation	14
3.5.3.1	Datasets and Ground Truth	14
3.5.3.2	Experimental Controls and Procedure	14
3.5.3.3	Evaluation Metrics	15
3.5.4	System-Level Evaluation	15
3.6	Analysis and Research-Question Answers	17
3.7	Evaluation Controls and Reproducibility	17
4	System Design and Implementation	19
4.1	System Scope and Use Cases	19
4.2	Design Requirements	20
4.3	Overall Architecture	20
4.4	Agent Orchestration and Workflow Execution	21
4.4.1	Agent Orchestration and Shared State	21
4.4.2	Workflow Execution	23
4.5	Specialised Engineering Capabilities	23
4.5.1	Database Query	23
4.5.2	Regulation and RAG	23
4.5.3	CAD Integration	25
4.5.4	Payload Calculation	25
4.5.5	Agility Analysis	29
4.6	User Interface and Human Interaction	29
4.7	Traceability, Error Handling, and Safety	29
4.8	Extensibility and Implementation Limitations	31
5	Results and Discussion	33
5.1	System Demonstration	33
5.1.1	User Interface Implementation	33
5.1.2	Single-Subagent Demonstrations	33
5.1.2.1	database subagent	34
5.1.2.2	Regulation subagent + RAG	35
5.1.2.3	CAD subagent	35
5.1.3	Multi-Subagent Workflow Demonstrations	36
5.1.3.1	Database and Payload Workflow	37
5.1.3.2	Database, Agility, and CAD Workflow	38
5.2	Evaluation Design Overview	38
5.3	Evaluation Results and Analysis	41
5.3.1	Offline-level Evaluation	41
5.3.2	Component-level Evaluation	43
5.3.2.1	Orchestrator Routing	43
5.3.2.2	Database Question Answering	44

5.3.2.3	Regulation Retrieval	45
5.3.2.4	Regulation Answering Strategies	46
5.3.2.5	RAG Chunk-size Ablation	47
5.3.3	System-level Evaluation	48
5.3.3.1	End-to-End Workflow Execution	48
5.3.3.2	Repeated Latency for Payload-independent Tasks . .	49
5.3.4	Implementation-level Validation and Failure Diagnosis	49
5.4	Summary of Evaluation Findings	50
5.5	Answers to the Research Questions	50
5.5.1	RQ1: Architecture and Design	50
5.5.2	RQ2: Prototype Capability	51
5.5.3	RQ3: Performance Trade-Offs	51
6	Conclusion	53
6.1	Summary of the Research	53
6.2	Main Contributions	54
6.3	Limitations	55
6.4	Future Work	56
6.5	Concluding Remarks	57
	Bibliography	59

List of Figures

3.1	Research process and evidence stages. The three evaluation branches assess deterministic and interface checks, individual capabilities, and complete workflows, respectively. The stage names correspond to the sections below.	9
4.1	System architecture with one orchestrator, five subagents, and one non-agent RAG management node.	22
4.2	Workflow of the Database subagent for natural-language database querying. The agent interprets the user's request, generates and executes an SQL query using the local database schema, and processes the returned records into a structured response.	24
4.3	Workflow of the Regulation subagent. Regulatory files from the company intranet are organised into a local regulatory document library and processed into both a structured rule base and a persistent vector database. When a user submits a regulatory query, the Regulation subagent answers it through structured rule querying, RAG-based retrieval, or a combination of both, while preserving the document sources to support traceability.	26
4.4	Workflow of the CAD subagent. The agent interprets a natural-language CAD request, identifies the required operation and parameters, selects a predefined parametric template, and modifies its exposed dimensions through the CAD automation API. The resulting model is exported and stored, after which the file path is returned to the user and the generated file is made available through the system interface.	27
4.5	Workflow of the Payload subagent. The user provides layout-related parameters through the payload interaction panel, while vehicle parameters required for the calculation are retrieved by the Database subagent. These two input sources are combined by the Payload subagent, whose calculation logic is implemented as an independent calculator script based on the company-provided mathematical model. After the calculation, the system compares the payload results before and after layout adjustment and presents the changes in tabular form.	28
4.6	Workflow of the Agility subagent.	30

5.1	Implemented user interface of the multi-agent engineering prototype. The interface provides a conversational workspace, capability-specific task entry points, and structured views for engineering inputs and outputs.	34
5.2	Representative Database-subagent interaction showing the user request and the returned vehicle comparison table.	34
5.3	Representative Regulation-subagent interaction showing a regulatory question, the generated answer, and the supporting source information.	35
5.4	Representative single-subagent CAD operation. The user instruction shown in (a) was processed by the CAD subagent, which invoked the connected CAD automation function to produce the three-dimensional geometry shown in (b). The example documents successful execution of the selected operation but does not establish the reliability of CAD execution across different requests.	36
5.5	Representative Database-to-Payload workflow. The interaction panel in (a) is used to retrieve reference vehicle data and specify the required calculation inputs. These values are transferred to the Payload subagent, which applies the deterministic calculation routine and presents the resulting payload information in tabular form in (b).	37
5.6	Representative Database-to-Agility-to-CAD workflow. The interaction panel in (a) is used to retrieve reference vehicle data and specify the required agility parameters. After the parameters have been reviewed, the Agility subagent prepares the geometric inputs and the CAD subagent invokes the connected CAD automation function. The resulting agility-related three-dimensional geometry is shown in (b).	39
5.7	Correct first-action selection in 100 labelled routing requests, with ten prompts per expected outcome. Each bar is the percentage of prompts in that class routed to the labelled action; higher bars indicate more correct selections. Argument correctness is evaluated separately.	44
5.8	Database question answering across 100 scored requests, with 20 cases per question type. Blue bars show the percentage whose complete SQL result matched the reference result; orange bars show the percentage whose final answer matched the automatic answer target. Higher bars indicate more matches under the respective criterion; the two criteria need not agree.	45
5.9	Regulation retrieval on 104 annotated queries. Each point is the mean score when the top k retrieved chunks are inspected. Source Recall@ k measures annotated-source coverage, NDCG@ k rewards earlier relevant results, and FactHit@ k records retrieval of labelled answer facts. Higher values indicate better performance under that metric; the three measures have different meanings.	46

5.10	Retrieval scores for three token chunk sizes, with overlap fixed at 64 and the same 104 annotated queries evaluated for each setting. Bars show MRR (rank of the first relevant result), Source Recall@5 (annotated-source coverage in the top five), and FactHit@5 (labelled fact found in the top five). Higher scores are better on this corpus; settings are not tested on other document collections.	47
5.11	Pass rates for 50 labelled end-to-end workflows. Each bar is the percentage satisfying one criterion: exact tool set, prescribed sequence, required output, or overall task success. Higher bars indicate more passing workflows; overall success can coexist with extra tool calls. . .	48
5.12	Response-time distributions for 115 retained observations across 23 prompts, excluding Payload. Each pair of bars gives median and 95th-percentile time in seconds on a logarithmic horizontal axis; shorter bars indicate faster responses. CAD and Agility measure planning only, not external software execution.	49

List of Tables

3.1	Local development data and their use by the subagents.	10
3.2	Task-selection criteria for agent assistance.	11
3.3	Evaluation datasets and retained performance observations.	15
3.4	Evaluation metrics used across system components.	16
4.1	Design requirements and implementation mechanisms.	21
4.2	Main workflow-state fields and their roles in execution traceability. .	22
5.1	Overview of the evaluation experiments and their respective evaluation levels.	42
5.2	Offline validation results for deterministic components and evaluation assets.	43
5.3	Automatically evaluated regulation-answering results ($N = 104$ per entry).	46
6.1	Summary of research questions, evidence, and supported conclusions.	54

1

Introduction

This chapter establishes the background and motivation of this study, introduces the engineering problem addressed, and defines the overall direction and boundaries of the research. It also clarifies the intended contribution of the work and gives a brief overview of the thesis structure.

1.1 Background and Motivation

Heavy-duty trucks are complex engineering products composed of hundreds of components and systems, many of which have several variants. These combinations produce a large number of potential vehicle configurations. During truck development, engineers must therefore do more than select and assemble individual components. They must coordinate them to satisfy customer needs, functional requirements, regulatory constraints, and other engineering conditions. This coordination challenge is especially important during early-phase truck integration, before detailed vehicle development begins.

At Volvo, Geometrical Architects (GAs) are responsible for early-phase truck integration. Even a seemingly simple task, such as moving one component to create space for another, requires them to consider the placement of the modified component, effects on nearby parts, customer needs, regulatory limits, maintenance requirements, and vehicle performance. If a proposed design fails to satisfy one or more requirements, the GAs must develop and evaluate another solution. This iterative work requires them to retrieve and combine information from several sources while conducting multiple analyses, which makes early-phase vehicle integration both information-intensive and time-consuming.

This way of working creates two particularly important challenges. The first is fragmentation across data, knowledge, and tools. Engineering data are distributed across several database systems, so GAs must understand the terminology and query methods used by each system. This requirement also creates a barrier for less experienced engineers. Evaluating a truck configuration may require knowledge from several engineering domains, while individual calculations and analyses may rely on separate tools or support from other teams. Collecting and combining these resources adds substantial effort. The second challenge is repetition. Each alternative design solution requires similar calculations and analyses. Existing automation can support individual operations, but it does not necessarily integrate information and

actions across the different systems involved in the wider workflow. These limitations motivate the investigation of approaches that can reduce repetitive work and simplify access to distributed engineering resources.

Several artificial intelligence (AI) technologies are relevant to this context, including Large Language Models (LLMs), Retrieval-Augmented Generation (RAG), and AI agents. LLMs can process and generate natural language and have been applied in professional fields such as medicine and law [1, 2]. RAG gives an LLM access to external information and has been used in manufacturing quality control to retrieve knowledge for defect analysis and diagnosis [3]. By combining language-model reasoning with tool use and task execution, an AI agent can interact with external systems rather than only generate text. Multiple specialised agents can then be co-ordinated within a multi-agent system. Similar agent-based tools are already used in software engineering to generate, debug, and modify code.

AI methods have also been applied to vehicle design, autonomous driving, manufacturing, and maintenance [4]. Their application to early-phase vehicle integration remains less developed. Nevertheless, their use in other domains indicates potential for coordinating information access and bounded engineering operations in this setting.

1.2 Problem Definition

Applying AI to vehicle integration presents several technical challenges. Early-phase truck integration is not a homogeneous task. It includes information retrieval, regulatory analysis, customer-requirement analysis, engineering calculations, and interaction with specialised software. These activities use different data sources, tools, forms of expertise, and output formats. Their heterogeneity motivates this thesis to investigate a modular multi-agent architecture in which bounded capabilities can be developed and coordinated separately.

Three challenges are particularly relevant to this investigation.

First, heterogeneous data and knowledge must be integrated. Early-phase truck integration uses structured vehicle data, unstructured regulations, technical material, and tacit engineering knowledge. These information sources differ in format and meaning, which complicates their use within one workflow.

Second, an AI-based assistant must interact with several engineering tools. Truck-integration analysis is not limited to text reasoning. It may require access to databases, CAD software, numerical calculation tools, and document-management systems. These operations require explicit interfaces and execution controls rather than unrestricted text generation.

Third, the system must be extensible to heterogeneous tasks. Early-phase truck-integration scenarios may evolve as new analysis dimensions, data sources, and engineering tools are introduced. A tightly integrated architecture may require substantial changes when new capabilities are added. The system therefore needs explicit extension points that allow functional modules to be developed and integrated with-

out reconstructing the entire architecture.

In summary, there is limited evidence on how AI-based capabilities can be organised to integrate the heterogeneous data, domain knowledge, and engineering tools used in early-phase truck integration. This thesis therefore designs and evaluates a modular, tool-orchestrated multi-agent system with explicit mechanisms for traceability and extension. The study examines the implemented artifact and its bounded task set rather than claiming production-scale performance.

1.3 Research Aim

The aim of this thesis is to design, implement, and empirically evaluate a tool-orchestrated multi-agent system for a bounded set of early-phase heavy-duty truck-integration tasks. The system translates natural-language requests into traceable workflows across heterogeneous engineering resources and provides explicit extension points for adding capabilities. The study evaluates task execution, declared contracts and invariants, and performance trade-offs. It does not seek to replace existing engineering processes or professional engineering judgement.

1.4 Research Questions

The thesis addresses the following research questions:

- RQ1:** How can a tool-orchestrated multi-agent architecture integrate heterogeneous engineering resources for early-phase truck integration while supporting traceability and extensibility?
- RQ2:** To what extent can the developed prototype correctly route and complete its supported workflows and satisfy its declared contracts and invariants on the defined benchmark?
- RQ3:** What trade-offs in accuracy, evidence faithfulness, reliability, and latency arise across the evaluated tasks and system configurations?

1.5 Contributions

The main contributions of this thesis are:

1. **A modular and extensible multi-agent system.** A tool-orchestrated framework was developed to integrate AI agents with heterogeneous engineering data, domain knowledge, calculations, and selected engineering tools for a bounded set of early-phase truck-integration tasks.
2. **Integration of AI capabilities into engineering workflows.** The framework shows how specialised agents can be coordinated to support engineering information retrieval, requirements and regulatory analysis, engineering calculations, and selected CAD-related operations.

3. **Empirical evaluation of the developed system.** Reproducible evaluation methods were established to examine routing, Database QA, Regulation/RAG retrieval and answering, retrieval configuration, End-to-End workflow execution, deterministic Payload correctness, interface conformance, traceability, and component response time.
4. **Identification of strengths and limitations.** The evaluation characterises the prototype on the defined benchmark and identifies limitations and trade-offs relevant to the further development of AI-assisted engineering systems.

1.6 Scope and Delimitations

This thesis focuses on the application of AI-based methods to early-phase heavy-duty truck integration. It develops a modular and extensible multi-agent prototype that integrates engineering data, domain knowledge, and selected engineering tools to provide information and analytical support. Extensibility is examined through the implemented architecture and its explicit extension points. Production scalability is not evaluated. The system is positioned as an engineering assistant, while final decisions remain with engineers.

This study does not cover the complete truck-integration process. Activities that require substantial tacit engineering knowledge or complex engineering judgement are outside the scope. The system is not intended to design trucks autonomously, replace engineering judgement, or provide unrestricted CAD and three-dimensional reasoning capabilities. Production deployment and changes in engineer productivity are also outside the scope of the study.

1.7 Thesis Structure

This thesis is organised into six chapters. Chapter 2 introduces the theoretical background, including LLMs, RAG, AI agents, tool use, multi-agent systems, and their engineering applications. Chapter 3 describes the methods used to design, develop, and evaluate the proposed system. Chapter 4 presents the system architecture, implementation, and domain-specific capabilities. Chapter 5 reports and discusses the evaluation results in relation to the research questions. Chapter 6 summarises the main findings and contributions and outlines directions for future work.

1.8 AI Disclaimer

During the preparation of this thesis, generative AI tools were used for grammar checking, language editing, support in organising the written argument, and literature discovery. The authors reviewed and verified the resulting suggestions and all cited sources. The authors remain responsible for the content and conclusions presented in this thesis.

2

Theory

Following the introduction of the research problem and scope in Chapter 1, this chapter provides the theoretical background needed to understand the problem and technologies relevant to addressing it. It covers both engineering context of early-phase vehicle integration and the AI concepts underlying the proposed system, and concludes by relating these areas to the research gap addressed in this thesis.

2.1 Early-Phase Vehicle Integration and Requirements

Early-phase vehicle integration coordinates and evaluates the vehicle architecture, functional requirements, and key engineering constraints before detailed design begins. Unlike the design of an individual part, vehicle integration considers several systems and their interactions, which makes it inherently cross-domain [5].

In heavy-duty truck development, early integration requires engineers to consider customer needs, regulatory requirements, and vehicle configuration together. Spatial layout, load capacity, vehicle dimensions, and driving performance can influence one another [5]. Engineers must therefore evaluate individual parameters while also combining information from several sources to assess the vehicle concept as a whole [6].

The required information includes structured vehicle and configuration data, unstructured regulatory texts and technical knowledge, and outputs from engineering tools. Early-phase vehicle integration can therefore be viewed as a multi-domain activity involving heterogeneous information, stakeholder needs, and analysis tools. This context motivates the investigation of AI-based support for information access and bounded engineering tasks.

2.2 Large Language Models and Agentic Workflows

A Large Language Model (LLM) is trained on large text collections to process and generate language in response to instructions. Models such as GPT-4 have demonstrated broad language-processing capabilities in professional and academic tasks.

However, their outputs may contain factual and reasoning errors, so an LLM should not be treated as an authoritative engineering knowledge source [7].

Tool use allows an LLM to delegate operations that it cannot or should not perform through text generation alone. For example, a model may call a calculator, search system, database, or software interface and use the returned result in a later step. The model can select a tool and formulate its input, while the external function performs the bounded operation [8].

Agentic workflows extend this pattern to multi-step execution. The ReAct approach combines reasoning and acting so that a model can select an action, invoke a tool, observe the result, and decide what to do next [9]. In this thesis, an LLM-based agent is therefore understood as a software component in which a language model interprets a task and coordinates bounded tool calls towards a defined goal.

This pattern is relevant to engineering applications because it separates flexible natural-language interaction from deterministic computation and controlled access to data. The LLM can support request interpretation and coordination, while specialised tools remain responsible for calculations, retrieval, and external software operations.

2.3 Multi-Agent Systems and Orchestration

A multi-agent system contains several agents with distinct responsibilities that exchange information while completing a task. Dividing work among specialised agents can support modular implementation and limit each agent to a bounded set of tools. LLM-based multi-agent systems have been studied for complex task solving, collaboration, and simulation. Their main design concerns include role definition, task assignment, communication, and coordination [10].

Orchestration coordinates execution across the agents. It may determine the order of operations, assign subtasks, pass intermediate information, and combine returned results. In an engineering application, separate agents may be responsible for database access, knowledge retrieval, engineering calculations, or software operations, while an orchestrator coordinates these capabilities within one workflow [11].

Multi-agent systems also introduce communication overhead, coordination failures, and error propagation. Increasing the number of agents does not necessarily improve performance [12]. The decomposition of responsibilities must therefore be justified by the interfaces and task boundaries of the application. This thesis investigates one such design but does not assume that it is superior to every single-agent alternative.

2.4 Natural-Language Access to Structured Engineering Data

Vehicle-development information is often stored as structured data in internal systems. Accessing these data may require knowledge of the database schema, en-

engineering terminology, and the query language used by each system. Obtaining information can therefore require substantial system knowledge even when the underlying records already exist.

Text-to-SQL translates a natural-language question into an executable SQL query [13]. LLM-based Text-to-SQL methods can use the question together with database-schema information to generate a query, reducing the need for users to write SQL directly [14]. In this thesis, Text-to-SQL provides an interface between natural-language requests and structured vehicle data. Its usefulness depends on executable queries, appropriate result scope, and accurate interpretation of the returned values.

2.5 Retrieval-Augmented Generation for Regulatory Knowledge

Retrieval-Augmented Generation (RAG) retrieves information from an external collection and supplies it to an LLM during answer generation [15]. This approach can make relevant source material available to the model and can support source traceability. It is often used to reduce reliance on parametric model knowledge, but retrieval alone does not guarantee that every generated claim is correct or supported [16].

For regulatory analysis, a RAG system can index regulatory documents, retrieve passages relevant to an engineer’s question, and provide those passages to the LLM for synthesis. In this thesis, RAG is used to retrieve information from a local collection of project-related regulatory documents and to retain source information. Knowledge-base maintenance also allows the document index to be updated when files are added or revised.

2.6 Engineering Automation and Human Oversight

Engineering automation uses computer systems to perform repeatable or rule-based tasks that would otherwise require manual work. AI-based systems extend conventional automation with natural-language interpretation, information retrieval, and multi-step coordination. However, engineering use requires explicit scope, input conditions, and verification mechanisms so that errors can be detected and contained [17].

Traceability is important because engineers must be able to inspect the basis of a result. A system can support such inspection by retaining inputs, source documents, tool calls, intermediate results, and errors. For tasks involving uncertainty or engineering risk, human oversight allows an engineer to review, modify, or reject the system output. The responsibilities of the system and the engineer should therefore be distinguished clearly [18].

2.7 Synthetic Data for Confidential Engineering Systems

Industrial engineering data are often restricted by confidentiality and access controls. Available records may also be insufficient for systematic testing. Synthetic data can provide predefined structures and characteristics without exposing production records, creating a controlled environment for system development and evaluation [19]. Manufacturing research also uses synthetic data to address limitations in data access, privacy, and quality [20].

The design requirements and data structures in this study were informed by the industrial context. Quantitative evaluation used synthetic vehicle data rather than production records. The synthetic database represents the tables, relationships, and query patterns required by the prototype, but it is not assumed to reproduce the distribution or business semantics of the production environment. The resulting measurements therefore characterise behaviour on the defined benchmark rather than performance in unrestricted industrial use.

2.8 Synthesis and Research Gap

Existing research shows that LLMs, tool use, RAG, and multi-agent systems can support knowledge retrieval, database access, task automation, and multi-step execution [8, 9, 10, 15]. These capabilities are often studied separately or in applications with a narrower set of resources. Less evidence is available on how they can be combined within a traceable engineering assistant that accesses structured data, documents, deterministic calculations, and external engineering software.

For early-phase vehicle integration, such a system must coordinate operations across heterogeneous resources while allowing engineers to inspect the information and actions behind its outputs. Final engineering decisions must remain under human authority [18]. This thesis addresses that design problem by implementing and evaluating a tool-orchestrated multi-agent prototype for selected early-phase truck-integration tasks, with explicit attention to resource integration, traceability, extensibility, bounded execution, and human oversight.

3

Research Methodology

Building on the engineering context and technical background in Chapter 2, this chapter explains how the use cases were selected, the artifact was developed, and its behaviour was evaluated. Its sections follow the stages in Figure 3.1 and describe the controls used in the evaluation.

3.1 Research Design

3.1.1 Design-Science Case Study Strategy

This study adopts a design-science research strategy centred on a software artifact for early-phase truck integration. Design science emphasises problem relevance, artifact quality, and evaluation [21]. The study follows the stages of problem identification, objective definition, artifact development, demonstration, evaluation, and communication [24].

The research also has a case-study character because the requirements and capability boundaries originate from early-phase heavy-duty truck integration. This context defines the task types, data forms, and software interfaces. The evaluation examines the implemented prototype using declared datasets and workflow tests.

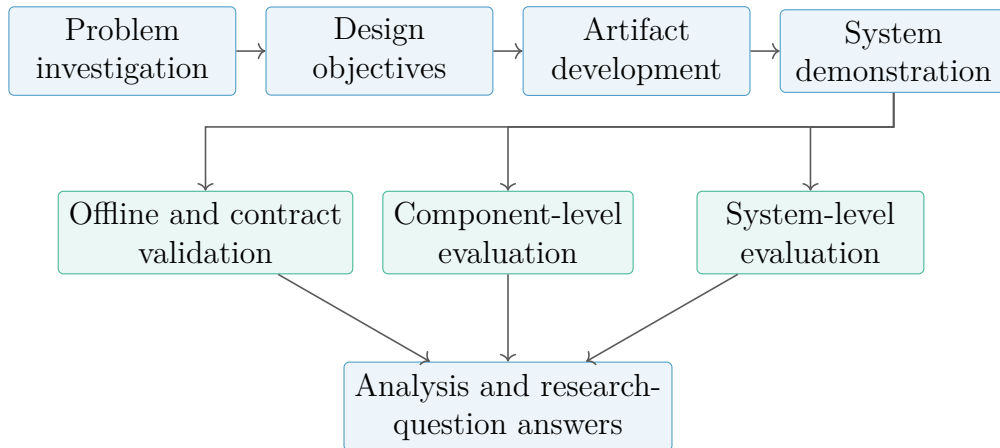


Figure 3.1: Research process and evidence stages. The three evaluation branches assess deterministic and interface checks, individual capabilities, and complete workflows, respectively. The stage names correspond to the sections below.

3.1.2 Industrial Context and Unit of Analysis

The industrial context is early-phase heavy-duty truck integration, where vehicle configuration, target market, regulatory limits, payload capacity, and geometric design must be considered together. The unit of analysis is one tool-orchestrated engineering workflow initiated by a user request. A workflow includes request interpretation, tool routing, specialist execution, intermediate results, and the final response.

Depending on the task, a workflow may invoke one tool or several tools in sequence.

3.2 Problem Investigation

The task scope was identified through discussions with the industrial supervisor and examination of the existing early-phase integration workflow described in Chapter 4. Load capacity and vehicle agility were selected as the core assessment tasks because both have defined calculation or geometric criteria. The workflow analysis identified the supporting needs for vehicle data, regulatory information, and CAD operations.

3.2.1 Development Data and Task Selection

During development, the local SQL database contained 100 truck configuration records and 20 BEP parameter records. The configuration records provide truck variants, function groups, and product types. The BEP records provide key vehicle packaging and geometric parameters. Table 3.1 shows how the subagents use these records and the other local data resources. Regulatory documents and CAD templates are maintained separately from the SQL database. For evaluation, the Database subagent used a simulated database generated from the development database (Section 3.4.1).

Table 3.1: Local development data and their use by the subagents.

Subagent	Local data used	How the data are used
Database	SQL truck configurations and BEP parameter records	Retrieves truck variants, function groups, product types, and BEP specifications.
CAD	Local parametric geometry templates; user-defined or Agility-supplied dimensions	Generates three-dimensional restriction surfaces for width, height, and water level.
Regulation	Local rule records and indexed regulatory documents for 13 countries	Retrieves rules and document passages to answer regulatory questions.
Agility	BEP geometric parameters retrieved from SQL; user-defined geometry values	Calculates agility geometry and passes surface parameters to CAD.
Payload	BEP vehicle parameters retrieved from SQL; user-defined layout values	Calculates changes in front- and rear-axle loads and gross vehicle weight after layout changes.

Natural language is used to interpret Database and Regulation requests. The decisive Payload calculation remains deterministic. Agility and CAD actions require restricted operations and timeouts. Engineers retain responsibility for design approval and formal compliance.

Table 3.2 summarises the criteria used to select tasks for the prototype.

Table 3.2: Task-selection criteria for agent assistance.

Criterion	Suitable characteristic	Design response
Input representation	A request can be translated into explicit parameters	Structured tool schemas and clarification
Verifiability	Output can be compared with data, rules, or calculations	Reference SQL, oracle calculations, and invariants
Operational risk	The action can be bounded and monitored	Restricted tools, timeouts, and process isolation
Traceability	Intermediate results can be retained	Trace identifier, node, tool, status, output, and error
Human responsibility	Final judgement remains reviewable	Evidence display and non-autonomous positioning

The selected workflows assign tasks and decisions as follows.

Bounded tool execution. A specified calculation or retrieval can be run through a restricted tool and checked against a reference result. The Payload calculation and vehicle-data query are examples in this prototype.

Assisted analysis. The system gathers evidence and calculates geometric quantities for Agility analysis. An engineer interprets the outputs in the design context.

Engineer-driven decision. The system supplies source material and calculations. Engineers make vehicle-architecture and compliance decisions.

3.2.2 Representing Workflows for Agent Execution

A selected task is represented through executable functions, structured data, and tools exposed through an integration layer.

Executable code. Deterministic functions for repeatable operations, such as an axle-load formula or a parameterised automation step. Repeatable and independently testable, but rigid: a change in the engineering procedure is a code change.

Structured parameters and data. Rules, limits, component types, and relationships held as data behind a stable interface. Scalable and inspectable, but bounded by what the schema anticipates, so a question outside it cannot be answered from this path.

Tools behind an integration layer. Approved functions, databases, and docu-

ment stores exposed to the model as typed, documented actions. This is the representation the prototype uses; the Model Context Protocol is one external standard for the same purpose [25]. An integration layer determines what the model may invoke, not whether the underlying data is correct.

Calculations are implemented in code, constraints are held in structured data, and the integration layer connects natural-language requests to the relevant tools. Each tool call and its result are recorded in the workflow trace.

3.3 Design Objectives

The problem investigation produced seven design objectives.

1. Provide one natural-language entry point that routes requests to the appropriate engineering capability.
2. Separate domain capabilities through restricted tools and subagents so that no specialist has unrestricted access to all underlying functions.
3. Retain intermediate outputs and source information for data queries, regulation retrieval, and calculated results.
4. Make deterministic calculations testable through an independent oracle, repeatability checks, and invariants.
5. Isolate external engineering-software operations through bounded commands, separate processes, and timeouts.
6. Permit extension through an additional tool, node, and routing entry without requiring other subagents to be rewritten.
7. Separate model errors, infrastructure errors, timeouts, and successful execution so that environmental failures are not reported as model errors.

3.4 Artifact Development

The system was developed iteratively, starting from the core engineering functions and gradually integrating them into the multi-agent architecture. The Database, Regulation, Payload, Agility, and CAD functions were implemented and tested individually before being exposed as tools to the agent system. These tools were subsequently integrated into the LangGraph-based workflow, together with the orchestrator and supporting components. Additional system-level functions, including the browser interface, project storage, execution traces, and evaluation scripts, were added as the implementation matured.

Development and evaluation were carried out iteratively. Deterministic functions and tool interfaces were checked before workflows involving the language model. Observed failures were classified by system layer, including the dataset, tool interface, routing logic, execution process, and evaluation procedure. Evaluation artifacts and raw outputs were versioned and retained to preserve experimental traceability.

The final evidence set was selected by checking each experiment’s dataset, execution path, scoring procedure, and raw report. It retains Routing, Database QA, Payload validation, synthetic-database checks, contract checks, Regulation/RAG retrieval and answering, chunk-size ablation, End-to-End workflow execution, and repeated latency for the selected task types.

3.4.1 Data and Confidentiality Strategy

Enterprise vehicle data and internal documents are subject to access and confidentiality constraints. The public evaluation therefore uses a simulated SQLite database generated on the basis of the development database. It preserves the tables, fields, relationships, and query patterns required by the experiments. Synthetic data support repeatable software testing [23, 22].

Regulation evaluation uses a curated local corpus with annotated questions and source files. Run provenance retains the dataset and database hashes, model configuration, random seed, and document configuration.

Frozen datasets were not modified during a formal run. Any change to prompts or labels required a new dataset version and manifest. This rule limits the risk of adapting a test set after observing its results.

3.5 System Demonstration and Evaluation

The implemented workflows are demonstrated through selected single-capability and combined use cases in Chapter 5. The examples show the requested operations, tool calls, intermediate results, and final outputs. The following subsections define the offline, component, and system evaluations.

3.5.1 Evaluation Design Overview

The evaluation is aligned with the three research questions. RQ1 is addressed through inspection of the architecture, resource interfaces, traceability mechanisms, and extension points. RQ2 is addressed through Routing and Database QA, deterministic Payload validation and contract checks, Regulation/RAG benchmarks, and End-to-End workflow execution. RQ3 is addressed through the difference among strict execution, value coverage, and answer accuracy, the RAG chunk-size ablation, and task-specific response-time distributions.

The evaluation uses offline, component, system, and implementation checks. Offline checks cover datasets, the simulated database, reference SQL, and the deterministic Payload calculation. Component tests cover Routing, Database QA, Regulation retrieval and answering, and retrieval configuration. System tests cover End-to-End workflows and latency for five task types. Implementation checks cover tool contracts, invariants, typed errors, and trace format.

3.5.2 Offline and Contract Validation

3.5.2.1 Levels of Verifiability

The evaluation uses a check suited to each output type. Payload calculations are compared with an independent oracle and fixed invariants. Database queries are compared with executable reference SQL and required values. Regulation retrieval is checked against annotated sources and facts in the top- k results. Workflow execution is checked through the selected tools, their order, and the required output. Natural-language answers are scored against the annotated answer targets.

3.5.2.2 Contract Conformance as Capability Evidence

Contract conformance evaluates whether the implemented tools behave according to their declared interfaces and execution rules. The final offline evaluation includes six checks: Payload input acceptance; the zero-change invariant, which verifies that outputs remain unchanged when no modification is required; deterministic output under repeated execution; data preconditions; typed error handling; and trace-format compliance.

Together, these checks assess whether the implementation conforms to its documented inputs, numerical invariants, error-handling behavior, and trace requirements. In addition, deterministic components are evaluated through agreement with independent reference results, repeatability across repeated runs, and the zero-change invariant.

3.5.3 Component-Level Evaluation

3.5.3.1 Datasets and Ground Truth

The quantitative evaluation uses frozen datasets and saved raw reports. Table 3.3 summarizes the retained datasets, evaluation targets, and corresponding ground truth.

Ground truth is defined according to the task. Model-driven tasks are evaluated against expected actions, outputs, sources, or workflow behaviour, while deterministic Payload calculations are compared with an independently implemented closed-form oracle. Latency is treated as an observed response-time measure rather than an accuracy metric.

The routing benchmark is class-balanced, with ten requests for each of ten expected first-action classes: eight tool actions, clarification, and no-tool response. This design supports direct comparison across routing categories. Twenty routing requests additionally include expected tool arguments.

3.5.3.2 Experimental Controls and Procedure

To improve reproducibility and maintain consistency across experiments, the retained evaluations use frozen datasets, a fixed random seed, and saved experiment manifests. Each run records the model configuration, timestamp, dataset hash,

Table 3.3: Evaluation datasets and retained performance observations.

Dataset	Size	Primary target	Ground truth
Routing	100	First action and arguments	Expected tool, clarification, or no-tool outcome
Database QA	100	Query and answer behaviour	Executable SQL, returned values, and answer targets
Payload validation	60	Deterministic calculation	Independent closed-form oracle
Regulation QA	104	Retrieval and answer behaviour	Expected sources, fact terms, and answer terms
RAG ablation	3×104	Chunk-size sensitivity	Same annotated queries for each configuration
End-to-End	50	Integrated workflow execution	Expected tools, order, and required output terms
Latency	115	Task-specific response time	Recorded successful executions

database hash, raw observations, and aggregated results. Outputs from different runs are stored separately to preserve experimental evidence and prevent accidental overwriting. Reported values are traced to a declared dataset, configuration, raw observation set, and scoring procedure.

For experiments involving live model execution, outcomes are classified as successful runs, model errors, infrastructure errors, or timeouts. Provider-side HTTP failures are treated as infrastructure errors rather than evidence of model failure. Reported accuracy metrics use denominators corresponding to the actual evaluated samples.

3.5.3.3 Evaluation Metrics

Different system components are evaluated using task-specific metrics. Table 3.4 summarises the main metrics and their meanings. Wilson 95% confidence intervals are reported for observed proportions where applicable.

3.5.4 System-Level Evaluation

End-to-End performance is evaluated using exact tool-set match, tool-order accuracy, required-output accuracy, and overall task success. Response time is reported for Routing, Database QA, and the Payload-independent repeated-performance subset using the median and 95th percentile. The median represents typical performance and is less sensitive to extreme values, while the 95th percentile characterises the slower tail of the retained observations.

Before metric calculation, each request is classified as a successful execution, model error, infrastructure error, or timeout. A model error is recorded when the model output cannot be processed correctly or the requested task is not completed. Infrastructure errors include failures originating from the provider, network, or execution environment, while a timeout is recorded when execution exceeds the predefined

Table 3.4: Evaluation metrics used across system components.

Component	Metric	Meaning
Routing	Accuracy	Proportion of requests routed to the correct first action.
	Macro-F1	Average F1 across action classes, reflecting balanced routing performance.
	Argument accuracy	Proportion of labelled tool arguments predicted correctly.
Regulation retrieval	Source Recall@ k	Proportion of relevant sources found within the top- k results.
	Precision@ k	Proportion of the top- k retrieved results that are relevant.
	MRR	Indicates how highly the first relevant result is ranked.
	NDCG@ k	Measures ranking quality, giving more weight to relevant results near the top.
	FactHit@ k	Indicates whether the top- k results contain the facts needed to answer the question.
Database QA	Strict execution accuracy	Proportion of query results that exactly match the reference result.
	Row-restriction accuracy	Measures whether the intended rows are retrieved.
	Value coverage	Measures whether the returned result contains the values needed to answer the question.
	Automatic answer accuracy	Proportion of final answers matching the expected answer target.

time limit.

The full frozen dataset is used as the denominator whenever possible. When an analysis is based only on scoreable or retained observations, the total, evaluated, and excluded counts are reported explicitly.

3.6 Analysis and Research-Question Answers

The final stage relates the measured outcomes to the research questions. Chapter 5 presents the results and design implications, and Chapter 6 synthesises the answers.

Proportion estimates are reported with Wilson 95% confidence intervals. Latency is summarised using the median and 95th percentile, with bootstrap confidence intervals where appropriate.

The statistical analysis reports each result with its dataset size, scoring procedure, and model configuration. Differences among component metrics are described within their respective task settings.

3.7 Evaluation Controls and Reproducibility

To support consistent evaluation, the experiments use frozen datasets, fixed configurations, saved run manifests, raw outputs, and predefined scoring rules. These artefacts provide traceability from each reported result to its input, execution configuration, and scoring procedure.

For confidentiality, the public evaluation uses a simulated vehicle database and a curated local regulation corpus rather than enterprise data. The retained code, datasets, configurations, and analysis scripts support reproduction of the reported local evaluation procedures.

4

System Design and Implementation

Based on the research methodology and design objectives established in Chapter 3, this chapter translates real engineering workflows into a concrete system design. It presents the developed prototype and how it supports the engineering tasks within the defined scope of this study.

4.1 System Scope and Use Cases

In Volvo's heavy-duty truck development, customer features are defined to describe the functions and performance requirements that a vehicle is expected to satisfy, and they therefore provide an important basis for assessment during early-phase vehicle integration. These features cover different functional and performance-related aspects and are typically evaluated across several engineering domains. Within the scope relevant to this study, four customer features were considered:

- **Load handling:** the vehicle's ability to support the loading, securing, transportation, and unloading of goods.
- **Suitability for bodywork:** the ability of a vehicle chassis to meet the installation and integration requirements of different body types.
- **Load capacity:** the amount of payload that a vehicle can carry under constraints related to total vehicle mass and axle loads.
- **Vehicle agility:** the vehicle's maneuverability under conditions such as turning, negotiating slopes, and operating in uneven or spatially constrained environments.

This study does not attempt to cover all customer features relevant to early-phase vehicle integration, and implementing all four identified features within the limited project scope would not have been feasible. Therefore, following discussions with the industrial supervisor, **load capacity** and **vehicle agility** were selected as the two core assessment tasks supported by the prototype. The main reason for this selection was that both tasks can be evaluated using relatively clear and quantifiable engineering criteria. Load capacity follows a well-defined calculation procedure, while vehicle agility can be assessed through measurable geometrical indicators, including ground clearance, approach angle, departure angle, and ramp angle. These

characteristics make the two tasks particularly suitable for the development and systematic evaluation of AI-agent-supported engineering workflows. In contrast, load handling and suitability for bodywork often depend more strongly on specific body types, application scenarios, component layouts, and engineering judgement, making it more difficult to establish unified and repeatable evaluation criteria within the scope of this study.

Based on the selected two core tasks, we consulted our industrial supervisor and combined the existing engineering workflow, identified the steps and required capabilities to complete these tasks. The system must be able to query vehicle configurations and geometrical parameters, retrieve relevant regulatory materials, calculate payload from vehicle and component parameters, and combine vehicle data with CAD operations for agility analysis. Its functional scope therefore includes data access, regulation retrieval, deterministic calculation, and interaction with engineering software.

These requirements define five main use cases: vehicle configuration and geometrical parameters queries, regulation queries, payload calculation, agility analysis, and CAD interaction. Payload calculation and agility analysis correspond directly to the selected core tasks, while the remaining use cases provide supporting capabilities. The capabilities may be used independently or combined, in a multi-step workflow, information or calculation results from one stage can be passed to a later stage. The five use cases define the functional boundary for the architecture and workflow design described below.

4.2 Design Requirements

Based on the five use cases identified in Section 4.1, we further analyze the common requirements of these functions at the system level and summarize them into four design requirements. Specifically, different uses cases involve different engineering resources such as databases, regulatory documents, calculations and CAD, so a unified interaction method is needed. At the same time, the system needs to combine real engineering data and deterministic calculation to support the reliability of the results. Different use cases have relatively independent task and tool requirements, requiring the system to be modular and scalable. In addition, considering that engineering decisions still need to be made by engineers, the system should retain manual control of parameters and final results. The design requirements and their implementation mechanisms are shown in Table 4.1. These requirements guide the architecture and implementation and provide a basis for evaluation. They are design targets rather than claims that every mechanism is complete or production-ready.

4.3 Overall Architecture

The system is implemented as a locally running engineering application. Engineers interact with it through a browser-based web interface. The orchestrator, subagents, local database, and engineering resources are deployed on the same workstation.

Table 4.1: Design requirements and implementation mechanisms.

ID	Requirement	Implementation mechanism
DR1	Integrate heterogeneous engineering resources behind one interaction boundary	Unified user interface, orchestrator, subagents, and tool interfaces
DR2	Ground responses in real engineering data and deterministic operations	Database queries, document retrieval, source references, and deterministic engineering calculations
DR3	Provide modular and extensible specialist capabilities	Multi-agent architecture with independent subagents, restricted tool sets, and explicit extension points
DR4	Reserve human control over engineering decisions	Visible results, editable parameters, and bounded predefined operations

This arrangement supports access to locally stored data and engineering software without requiring a direct connection to Volvo’s production database.

This study adopts a multi-agent architecture that coordinates multiple specialist agents with an LLM-driven orchestrator. This architecture is selected mainly because the identified use cases in Section 4.1 involve different types of engineering resources and functions. Compared with using a single agent to directly manage all tools, dividing these use cases into relatively independent specialist agents can establish clearer responsibilities and tool boundaries, and enable each use case to be independently expanded and maintained. Therefore, this study adopts a multi-agent architecture mainly to achieve modular functionality and clear tool access boundaries. Based on this design, the system contains five subagents corresponding to the use cases defined in Section 4.1: Database, Regulation, Payload, Agility, and CAD. Each subagent connects to bounded engineering data, knowledge resources, or tools and interacts with the orchestrator through a common interface.

Figure 4.1 shows the overall architecture. Engineers can submit requests to the LLM-driven orchestrator through the user interface. The orchestrator interprets each request and coordinates the relevant subagents to perform a single-step or multi-step task.

4.4 Agent Orchestration and Workflow Execution

4.4.1 Agent Orchestration and Shared State

The orchestrator coordinates the subagents. After an engineer submits a request, it interprets the task and selects the required operation and subagent. It can respond directly when no engineering operation is required. Otherwise, it dispatches the task to a subagent through a structured tool call.

After a subagent completes its operation, it returns the result to the orchestrator. The orchestrator then either composes a response or invokes another subagent. A

4. System Design and Implementation

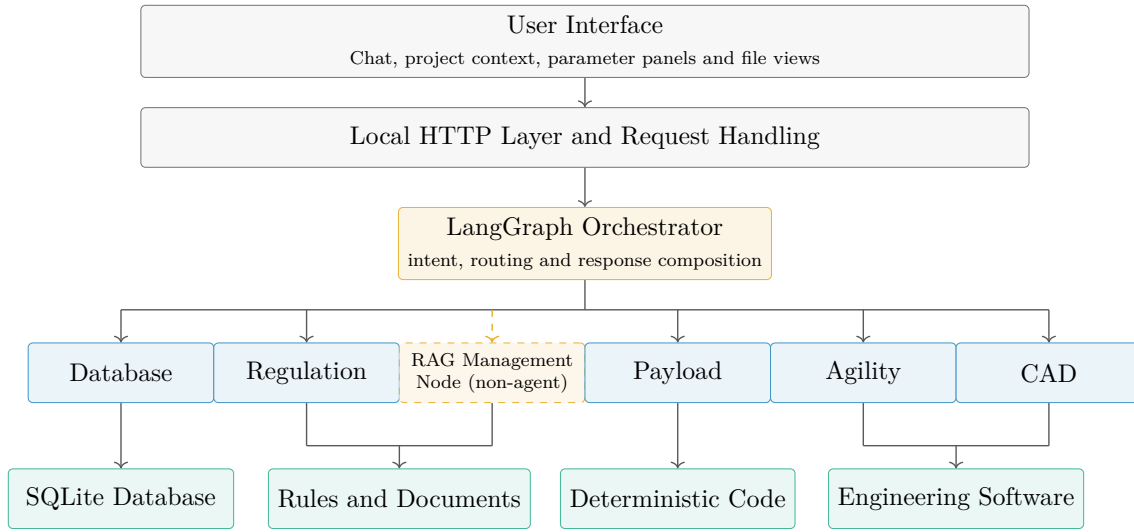


Figure 4.1: System architecture with one orchestrator, five subagents, and one non-agent RAG management node.

request may therefore require one subagent or a sequence of subagents. Later stages can use data or calculation results returned by earlier stages.

The system maintains a shared workflow state containing messages, tool interactions, intermediate results, and errors. Each request receives a unique trace identifier that associates its execution steps. Table 4.2 summarises the principal fields used for execution and traceability.

Table 4.2: Main workflow-state fields and their roles in execution traceability.

Field	Update behaviour	Purpose
<code>messages</code>	Appended and de-duplicated by message ID	Conversation, tool calls, and tool responses
<code>intermediate_steps</code>	Appended across specialist returns	Structured tool outputs and source extraction
<code>current_tool_call</code>	Supplied to one dispatched branch	Tool name, call identifier, and arguments
<code>trace_id</code>	Stable for one top-level request	Correlation of messages, steps, and failures
<code>errors</code>	Appended as structured records	Node, tool, exception type, message, and traceback

When a subagent returns a result, control passes back to the orchestrator, which selects the next step from the accumulated state. A subagent does not directly access the tools assigned to another subagent. This design supports multi-step execution while limiting the tool-access scope of each subagent.

4.4.2 Workflow Execution

Workflow execution has three stages: task assignment, engineering operation, and result integration. First, the orchestrator selects one or more subagents from the user's request. Each selected subagent then invokes its bounded resources or tools and returns the result to the workflow state. Finally, the orchestrator either invokes the next required subagent or consolidates the available results for the engineer.

For a multi-step task, the result returned by one subagent can become an input to the next, allowing several engineering capabilities to be combined within one workflow.

For example, a request to check whether a truck satisfies an EU vehicle-width restriction may require the Database subagent to retrieve the vehicle dimensions and the Regulation subagent to retrieve the applicable limit. The orchestrator then combines the returned information in the final response.

The orchestrator selects an execution path for the current request rather than applying one fixed workflow to every task. This permits different capabilities to be combined while preserving the functional boundaries of the subagents.

The system coordinates information and tool calls, while engineers remain responsible for interpreting the results and making engineering decisions.

4.5 Specialised Engineering Capabilities

4.5.1 Database Query

The Database subagent provides a natural-language interface to the local database. Engineers can describe a required vehicle configuration or parameter, and the subagent returns structured results such as tables. This interface reduces the need to formulate database-specific queries manually.

The prototype does not connect directly to Volvo's internal engineering databases. During development, authorised configuration information for 100 trucks and the geometric parameters required for payload and agility functions were organised in a small local SQL database. Quantitative evaluation used a separate simulated database, as described in Section 3.4.1.

The Database subagent uses the LangChain SQL Agent architecture. As Figure 4.2 shows, it interprets a natural-language request, generates an SQL query from the request and database schema, executes the query against the local database, and formats the returned records. Engineers can therefore retrieve data without writing SQL directly.

4.5.2 Regulation and RAG

The Regulation subagent retrieves information from a local regulatory knowledge base. Because confidentiality requirements restrict external network access, project-related regulatory files were organised in a local document collection. The collec-

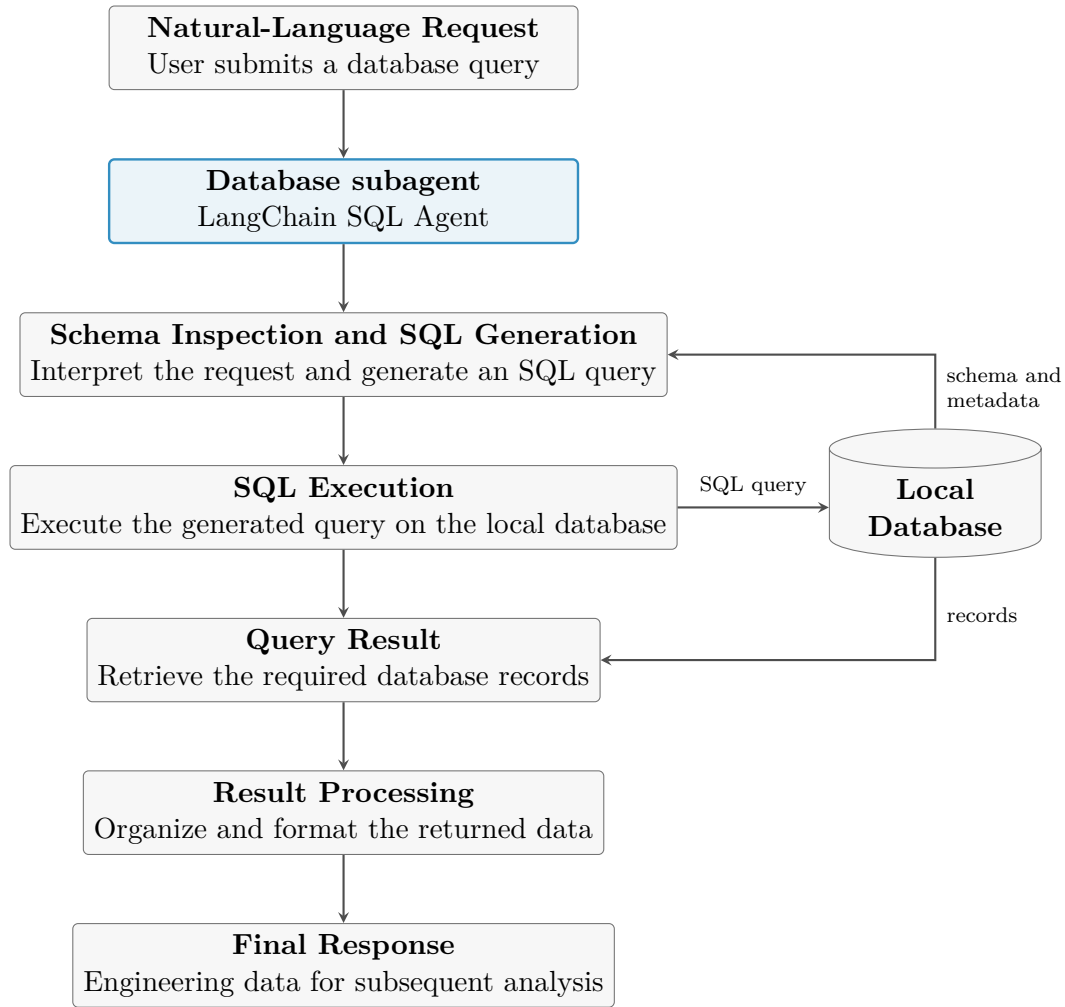


Figure 4.2: Workflow of the Database subagent for natural-language database querying. The agent interprets the user’s request, generates and executes an SQL query using the local database schema, and processes the returned records into a structured response.

tion covers requirements concerning vehicle dimensions, weight, payload, and agility across 13 countries.

The documents are divided into chunks, converted into vector representations, and stored in a persistent vector database. Figure 4.3 shows the Regulation-subagent workflow. For a regulatory request, the subagent retrieves relevant chunks, extracts information related to the question, and returns an answer with source information. Clearly structured limits can also be accessed through structured rules. The system can use the structured-rule and RAG paths separately or together, depending on the request.

This design gives the LLM access to project-related regulatory material rather than relying only on its parametric knowledge. It can reduce the manual work required to locate and organise documents. When files are added or revised, the local knowledge base can be reprocessed without retraining the language model.

4.5.3 CAD Integration

The CAD subagent provides the system with the ability to interact with CAD software, which is mainly used to support vehicle agility analysis. Because automation interfaces and API support differ among CAD platforms, the implementation uses predefined parametric templates rather than unrestricted free-form modelling. The templates represent length, width, height, water-level, and other limit surfaces related to vehicle agility, with the required dimensions exposed as editable parameters.

Figure 4.4 shows the workflow of the CAD subagent. When the user submits a CAD operation request, the CAD subagent first parses the natural-language instruction and identifies the operation type and relevant parameters. The agent then opens the corresponding template and modifies its parameters through the CAD automation API. After the operation is completed, the system exports the generated model, saves it to the specified location, and returns the file path to the user. The generated file is also displayed in the navigation area of the user interface.

The implementation primarily supports vehicle-agility analysis. Because it uses predefined parametric templates, it can also perform other bounded modelling operations when an appropriate template is available. The supported scope is limited to exposed parameters and predefined procedures; it does not provide unrestricted three-dimensional modelling.

4.5.4 Payload Calculation

The Payload subagent performs payload calculations through an independent calculator script based on the company-provided mathematical model. Users enter layout-related parameters through a structured interface, while vehicle parameters are retrieved by the Database subagent. Figure 4.5 shows how these inputs are combined. The Payload subagent compares the results before and after a layout adjustment and presents the changes in a table.

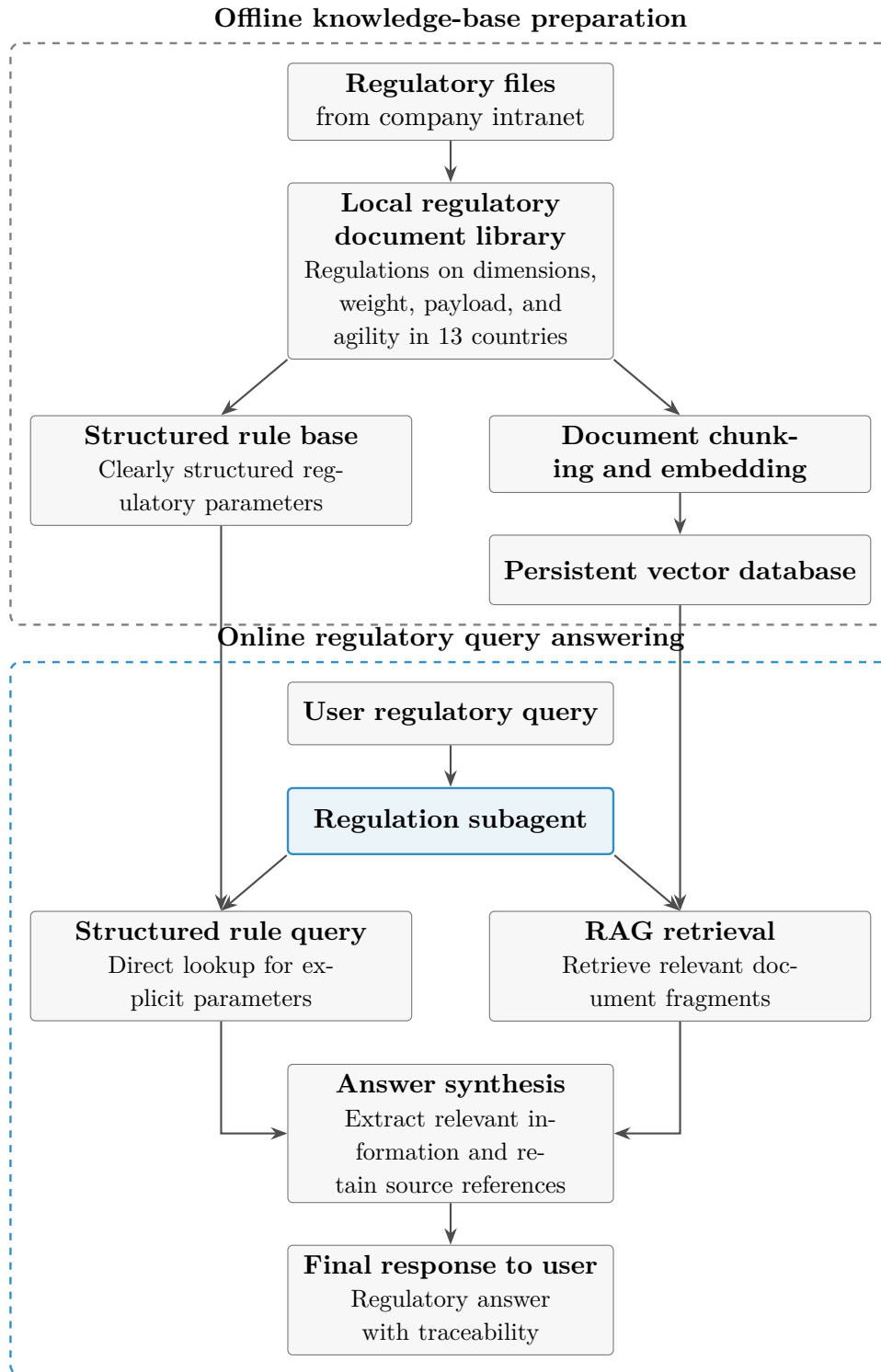


Figure 4.3: Workflow of the Regulation subagent. Regulatory files from the company intranet are organised into a local regulatory document library and processed into both a structured rule base and a persistent vector database. When a user submits a regulatory query, the Regulation subagent answers it through structured rule querying, RAG-based retrieval, or a combination of both, while preserving the document sources to support traceability.

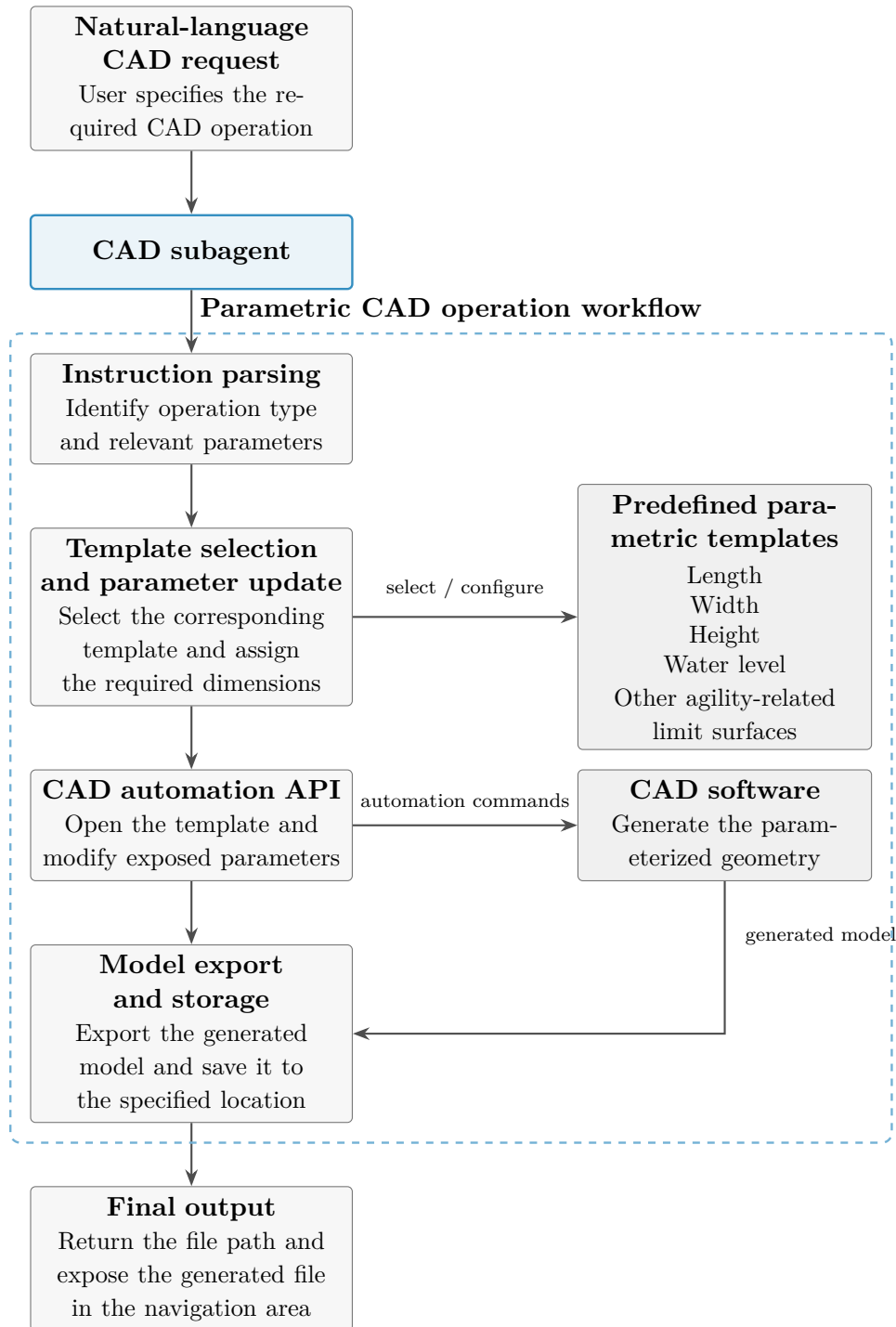


Figure 4.4: Workflow of the CAD subagent. The agent interprets a natural-language CAD request, identifies the required operation and parameters, selects a predefined parametric template, and modifies its exposed dimensions through the CAD automation API. The resulting model is exported and stored, after which the file path is returned to the user and the generated file is made available through the system interface.

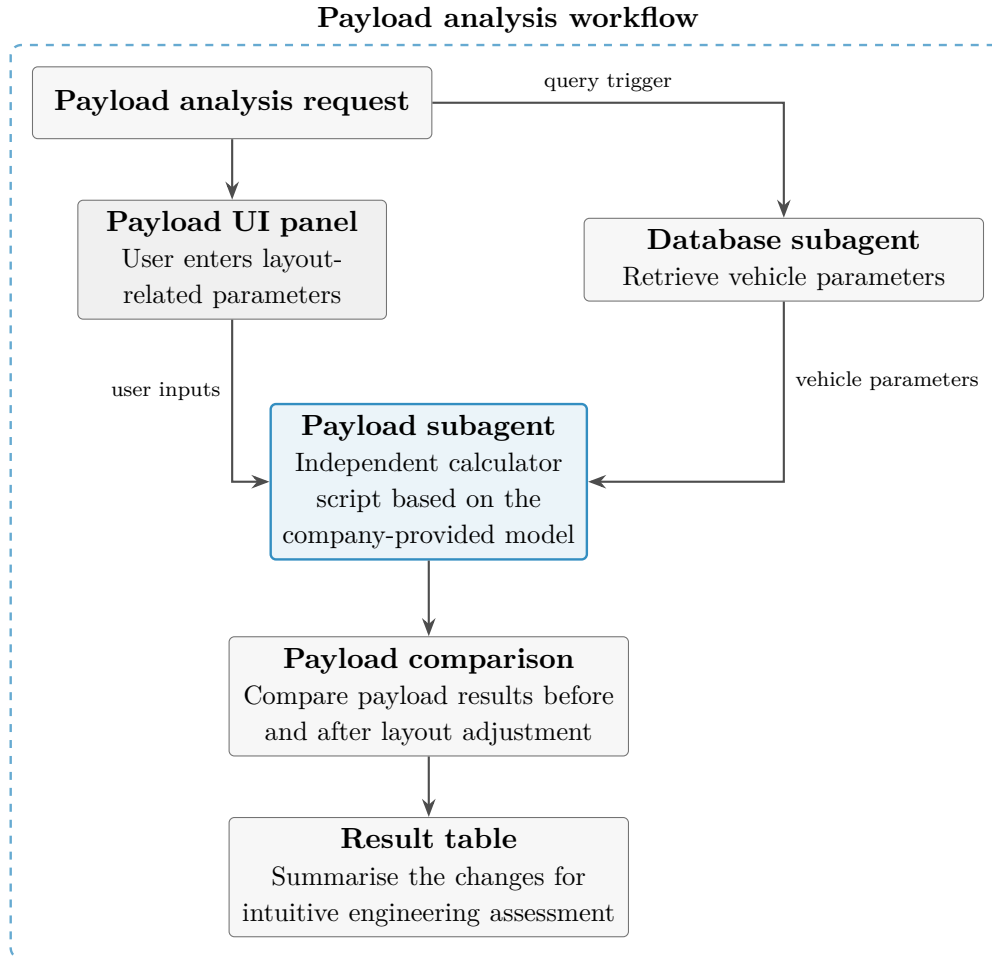


Figure 4.5: Workflow of the Payload subagent. The user provides layout-related parameters through the payload interaction panel, while vehicle parameters required for the calculation are retrieved by the Database subagent. These two input sources are combined by the Payload subagent, whose calculation logic is implemented as an independent calculator script based on the company-provided mathematical model. After the calculation, the system compares the payload results before and after layout adjustment and presents the changes in tabular form.

4.5.5 Agility Analysis

The Agility subagent supports vehicle-agility analysis. As shown in Figure 4.6, users provide a truck identifier and optional custom parameters through a structured interface. The Database subagent retrieves the corresponding geometric parameters. The Agility subagent then performs deterministic parameter calculations and passes the resulting values to the CAD subagent, which generates and exports the agility-related limit-surface model.

4.6 User Interface and Human Interaction

Engineers interact with the locally running system through a browser-based interface. A conversational workspace is combined with structured parameter panels, project information, dialogue history, file management, and dedicated Payload and Agility views.

For engineering tasks that require clear input parameters, users can view and modify relevant parameters through the parameter panel and submit calculation or analysis requests. For more open tasks, users can make requests directly through the dialogue interface, and the orchestrator will coordinate the corresponding subagents according to the task needs. The two interaction methods use the same underlying engineering calculation and data processing functions to maintain the consistency of the results. The system retains the necessary intermediate results, files and information sources during the execution process, so that engineers can view the generated results and conduct further checks.

The interface combines natural-language interaction with structured engineering operations. Engineers can select an interaction form suited to the task while accessing data, calculations, and tools through one application.

4.7 Traceability, Error Handling, and Safety

Each request receives a unique trace identifier. The workflow state records execution status, tool calls, and subagent outputs. Regulatory queries retain document sources, while calculations and CAD operations retain numerical outputs and generated files. These records allow engineers to inspect the basis and execution path of a result.

Tool-execution errors are recorded explicitly. If a required input is missing, a value has an invalid format, or execution fails, the system returns a clarification request or structured error rather than presenting the outcome as a valid result.

To limit unsupported generation, regulatory responses can use retrieved documents, while numerical results are produced by deterministic functions rather than generated directly by the LLM. When parameters or evidence are insufficient, the system requests additional information or reports that the task cannot be completed. Inputs are also checked for basic format and range before execution.

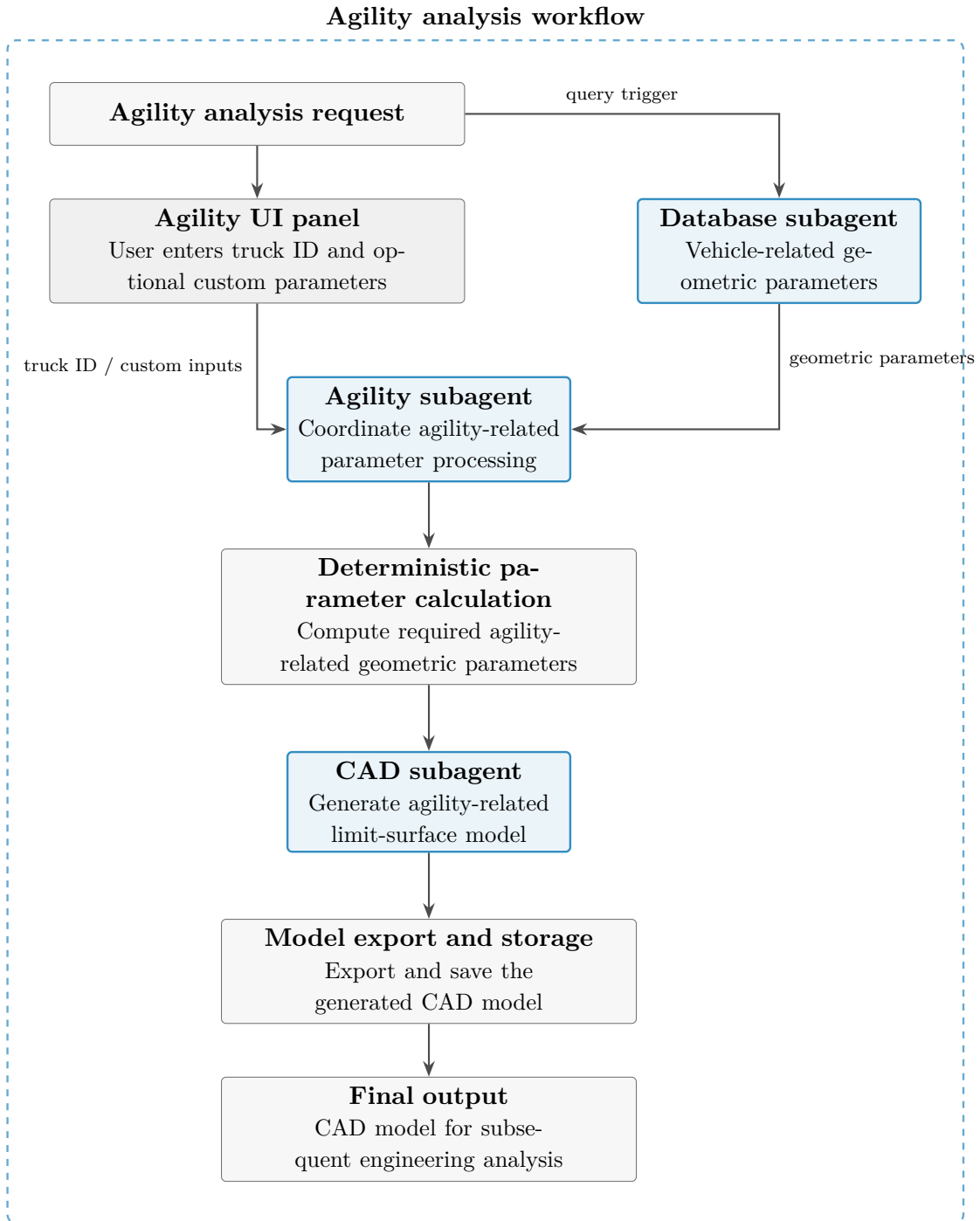


Figure 4.6: Workflow of the Agility subagent. The user provides the truck ID and optional custom parameters through the agility interaction panel, while vehicle-related geometric parameters are queried by the Database subagent. The Agility subagent combines these inputs and performs deterministic calculations to derive the parameters required for agility analysis. The resulting parameters are then passed to the CAD subagent, which generates the corresponding agility-related limit-surface model. The generated CAD model is exported and saved for subsequent engineering analysis.

These mechanisms improve traceability and constrain model behaviour, but they do not guarantee that every result is correct. The system is therefore positioned as an engineering-assistance tool, and engineers remain responsible for reviewing its outputs.

4.8 Extensibility and Implementation Limitations

The modular architecture provides explicit extension points. Adding a capability normally requires a tool interface, an implementation, registration with the orchestrator and relevant subagent, and corresponding tests. This structure limits dependencies among existing capabilities, although the effort required for extension was not evaluated experimentally.

The system remains a research prototype. It runs in one local environment and uses a locally maintained regulatory knowledge base, limited-scale structured data, bounded CAD operations, and the model services available during development. It does not provide the permission management, approval controls, operational monitoring, or failure-recovery mechanisms required for production deployment. These limitations inform the evaluation in Chapter 5 and the future work in Chapter 6.

5

Results and Discussion

This chapter presents the results obtained from the implemented prototype in Chapter 4 through system demonstrations and empirical evaluation. The findings are analyzed in relation to the supported capabilities, system performance, and the research questions defined in Chapter 1.

5.1 System Demonstration

Chapter 4 described the architecture, workflow design, and implementation mechanisms of the proposed multi-agent system. This section presents the implemented prototype from the user’s perspective and provides representative examples of its observable outputs. The purpose is to demonstrate that the designed interaction paths and engineering functions were implemented in an operational system. The examples in this section provide qualitative implementation evidence. They do not establish the accuracy, reliability, or generalisability of the system, which are examined separately in Sections 5.2 and 5.3. The prototype supports two forms of interaction. First, users can submit engineering questions through a conversational interface, after which the orchestrator selects the relevant specialist capability. Second, structured panels are provided for tasks that require explicit engineering parameters, such as payload calculation and agility analysis. This combination allows natural-language interaction while preserving structured inputs for calculation-intensive tasks.

5.1.1 User Interface Implementation

Figure 5.1 shows the implemented browser-based interface. The main workspace provides a conversational input area, task entry points for the available engineering capabilities, and panels for displaying intermediate or final outputs. The interface also provides access to project files and task status information. These elements allow the user to inspect the information used by the system rather than receiving only an unstructured final response.

5.1.2 Single-Subagent Demonstrations

The prototype first supports requests that can be completed by a single specialist subagent. Three examples are presented to demonstrate the different output forms

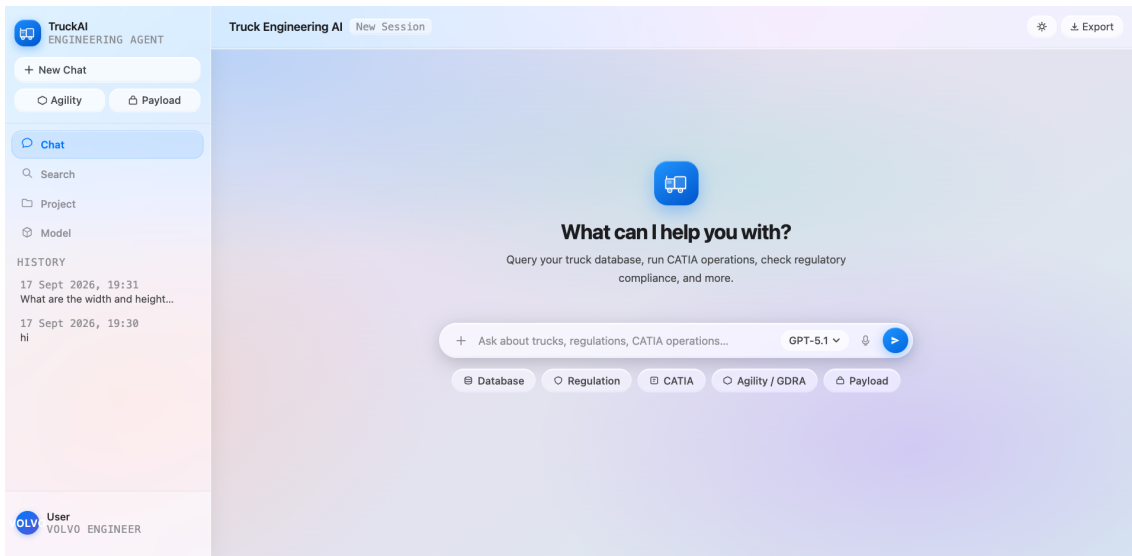


Figure 5.1: Implemented user interface of the multi-agent engineering prototype. The interface provides a conversational workspace, capability-specific task entry points, and structured views for engineering inputs and outputs.

implemented in the system. The Database subagent returns structured vehicle data, the Regulation subagent produces source-supported regulatory answers, and the CAD subagent executes an explicitly requested geometric operation. These examples demonstrate the observable behaviour of the individual capabilities.

5.1.2.1 database subagent

Figure 5.2 presents a representative Database request in which the user asks the system to compare selected vehicle specifications. The request is routed to the Database subagent, which queries the vehicle database and returns the requested parameters in a structured comparison table. Presenting the values in tabular form makes the selected vehicle records and parameter differences directly inspectable by the user.

Attribute	OM 24	OM 23
Wheelbase	WB5200	WB5200
Equivalent Wheelbase Turning	EWT5200	EWT5200

Figure 5.2: Representative Database-subagent interaction showing the user request and the returned vehicle comparison table.

5.1.2.2 Regulation subagent + RAG

Figure 5.3 shows a representative regulatory question processed by the Regulation subagent. The subagent retrieves relevant content from the regulatory knowledge base and uses the retrieved material to formulate its response. Both the answer and its supporting source information are presented to the user, allowing the basis of the response to be inspected.

You 16:57
Check compliance: length 18 m, width 2.55 m, height 4 m, weight 45 t

TruckAI 16:57
I'll check that against the regulations. Which market or region should I test compliance for? (e.g. EU, RUSSIA, AUSTRALIA, JAPAN, CN, INDIA, BRAZIL, CANADA, MEXICO, US, SA, KR, AR — or ALL for a full comparison.)

You 16:57
EU

TruckAI 16:57
Here's the EU compliance result for **length 18 m · width 2.55 m · height 4 m · 45 t**:

Dimension	Your Value	EU Limit	Result
Length	18 m	18.75 m (road train) / 16.50 m (articulated) / 12.00 m (motor vehicle/trailer)	NOTE
Width	2.55 m	2.55 m (2.60 m for conditioned/refrigerated superstructure)	PASS
Height	4 m	4.00 m	PASS
Weight	45 t	40 t (5/6 axles, general) / 44 t (5/6 axles, 40' ISO containers)	FAIL

- **Width — PASS:** exactly at the 2.55 m general EU limit.
- **Height — PASS:** exactly at the 4.00 m EU limit.
- **Length — NOTE:** only permissible if configured as a **road train** (limit 18.75 m). It **exceeds** the limits for an articulated vehicle (16.50 m) and for a motor vehicle/trailer alone (12.00 m).
- **Weight — FAIL:** 45 t exceeds the general 40 t EU maximum for 5/6-axle combinations, and also exceeds the 44 t allowance for 40' ISO container combinations. No standard EU category permits 45 t — a special permit would be required.

Additional notes:

- Semitrailer length has a documented discrepancy between sources (12.00 m vs. 13.59 m per Directive 96/53/EC).
- Axle-weight limits were **not** checked (single axle 10.0 t, drive axle 11.5 t) — please supply your axle configuration for a complete assessment.

Want me to run the same values against other markets, or check the axle-weight limits?

Sources: [Weight and Dimensions Summary China.pdf](#), [Weight and Dimensions Summary EU.pdf](#), [EU States-weights-2021-0.pdf](#), [EU.md](#), [Weights and Dimensions Directive.pdf](#)

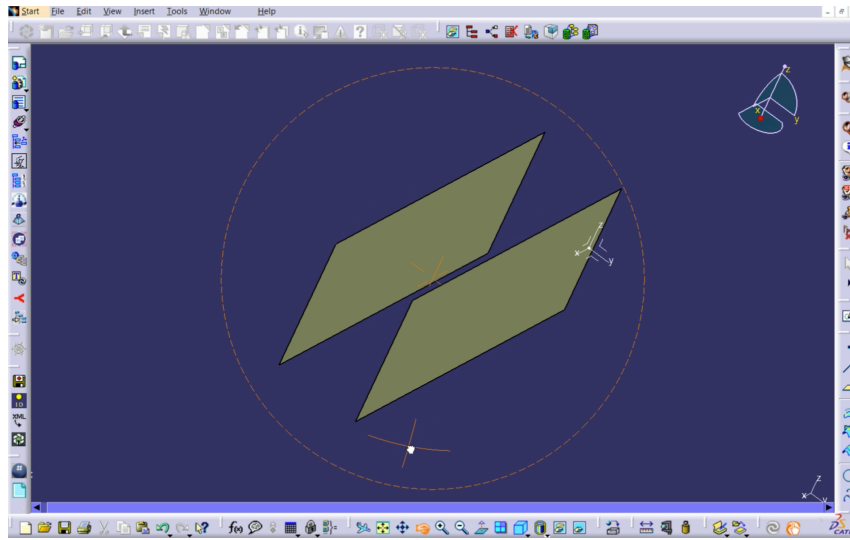
Figure 5.3: Representative Regulation-subagent interaction showing a regulatory question, the generated answer, and the supporting source information.

5.1.2.3 CAD subagent

Figure 5.4 presents a request handled directly by the CAD subagent. In this example, the user specifies a geometric operation, and the subagent translates the request into the corresponding CAD action. The resulting geometry is displayed in the CAD environment.



(a) User instruction submitted to the CAD subagent.



(b) Three-dimensional geometry produced in the CAD environment.

Figure 5.4: Representative single-subagent CAD operation. The user instruction shown in (a) was processed by the CAD subagent, which invoked the connected CAD automation function to produce the three-dimensional geometry shown in (b). The example documents successful execution of the selected operation but does not establish the reliability of CAD execution across different requests.

5.1.3 Multi-Subagent Workflow Demonstrations

In addition to requests handled by a single subagent, the prototype supports workflows in which information retrieved by one subagent is used by another capability. Two representative workflows are presented in this section. The first combines Database retrieval with Payload calculation. The second combines Database retrieval, Agility parameter preparation, and CAD execution. These examples focus on the observable transfer from reference data and input parameters to the resulting engineering output.

5.1.3.1 Database and Payload Workflow

Figure 5.5 presents a representative workflow involving the Database and Payload subagents. The upper panel shows the interaction panel used to retrieve reference vehicle data and specify the parameters required for the payload calculation. The Database subagent provides the relevant vehicle information, while the user can review or complete the calculation inputs before execution.

The selected reference data and input parameters are subsequently passed to the Payload subagent. The Payload subagent applies the deterministic calculation routine and returns the results in a structured table, as shown in the lower panel. The table separates the calculation outputs from the reference values and user-provided inputs, allowing the basis of the result to be inspected. The example demonstrates that vehicle data retrieved through the Database capability can be incorporated into the Payload calculation workflow without requiring the user to re-enter all reference parameters manually.

(a) Interaction panel for retrieving reference vehicle data and specifying the payload calculation inputs.

		FAL [kg]	RAL [kg]	Total [kg]	Load Capacity [kg]	Limiting factor	Status
Reference Truck: OM		6552	4075	10627	18089	GVW18	—
Project weight delta	Part 1 (Add)	+ 42	+ 8	+ 50	+ 31	—	18000 ✓
	Part 2 (Remove)	- 22	- 8	- 30		—	
	Part 3 (Adjust)	- 22	- 8	- 30		—	
	Tot	- 2	- 8	- 10		—	
PTV		6550	4067	10617	18120	GVW18	✓

(b) Payload calculation results presented in a structured table.

Figure 5.5: Representative Database-to-Payload workflow. The interaction panel in (a) is used to retrieve reference vehicle data and specify the required calculation inputs. These values are transferred to the Payload subagent, which applies the deterministic calculation routine and presents the resulting payload information in tabular form in (b).

5.1.3.2 Database, Agility, and CAD Workflow

Figure 5.6 presents a representative workflow involving the Database, Agility, and CAD subagents. The upper panel shows the interaction panel used to retrieve reference vehicle data and specify the geometric parameters required for the agility task. The retrieved values can be reviewed and supplemented by the user before the geometric operation is initiated.

The confirmed parameters are then processed by the Agility subagent to prepare the geometric inputs required for the analysis. The CAD subagent invokes the connected CAD automation function using these inputs. The resulting agility-related three-dimensional geometry is displayed in the CAD environment, as shown in the lower panel.

Together, the two examples demonstrate two forms of multi-subagent execution. In the Database-to-Payload workflow, retrieved vehicle data are used as inputs to a deterministic numerical calculation. In the Database-to-Agility-to-CAD workflow, the retrieved data are converted into geometric inputs and executed through an external CAD environment. The examples therefore document the implemented integration between the specialist capabilities.

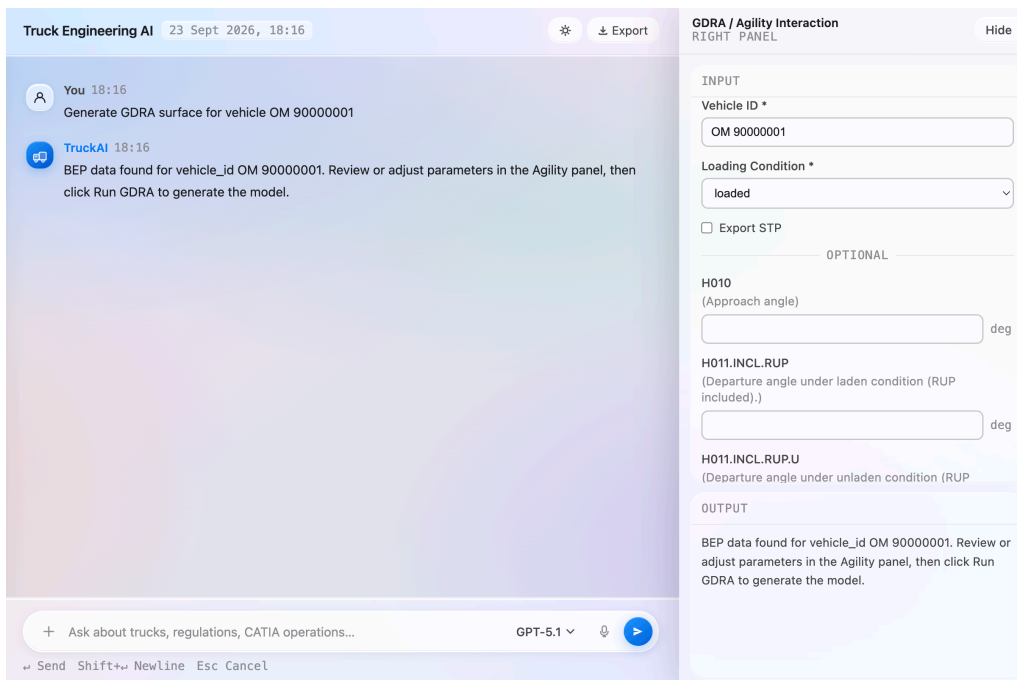
5.2 Evaluation Design Overview

Before an agent-based engineering workflow can be relied upon, both its tool-use behaviour and the outputs produced through that behaviour must be evaluated systematically. In this context, the primary concern is not the linguistic fluency of the underlying language model, but whether the system performs the appropriate engineering actions for a given request. These actions include selecting the correct subagent, generating an appropriate SQL query, retrieving relevant evidence, passing valid parameters to external tools, and producing an answer that can be traced back to the underlying computation or information source.

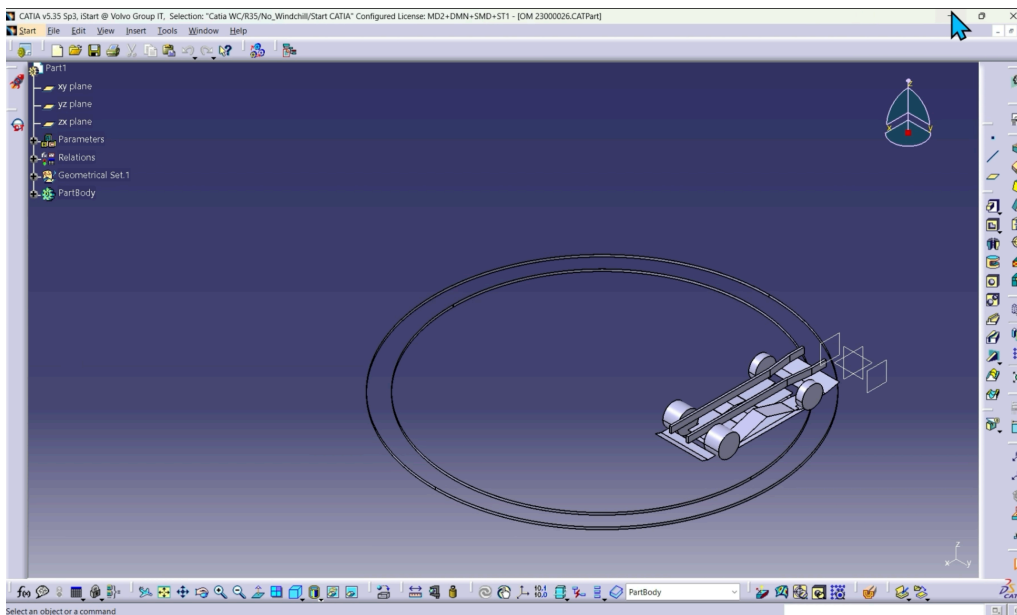
The outputs of the system can therefore be considered at several levels of verifiability:

- **Deterministic:** the result can be compared with an exact value obtained through an independently reproducible computation.
- **Reference-based:** the result can be evaluated against a predefined expected answer or a curated ground-truth dataset.
- **Traceable:** the result is accompanied by the query, retrieved document, tool output, or other intermediate evidence from which it was derived, allowing the result to be inspected directly.
- **Unvalidated:** the result lacks a reliable reference, traceable source, or reproducible calculation and therefore cannot be independently verified.

From an engineering perspective, greater confidence can be placed in outputs that fall within the first three categories. Consequently, deterministic operations are implemented as explicit computations wherever possible, while generated answers are expected to retain links to the queries, retrieved evidence, or tool outputs from



(a) Interaction panel for retrieving reference vehicle data and specifying the agility-related input parameters.



(b) Agility-related three-dimensional geometry produced in the connected CAD environment.

Figure 5.6: Representative Database-to-Agility-to-CAD workflow. The interaction panel in (a) is used to retrieve reference vehicle data and specify the required agility parameters. After the parameters have been reviewed, the Agility subagent prepares the geometric inputs and the CAD subagent invokes the connected CAD automation function. The resulting agility-related three-dimensional geometry is shown in (b).

which they originate. This distinction between generated language and verifiable execution is central to the evaluation design adopted in this study.

The proposed multi-agent system is not treated as a single language model whose performance can be adequately characterised by one overall accuracy value. Instead, it is considered as an engineering execution chain composed of several stages, including request routing, tool invocation, information retrieval, deterministic computation, and result synthesis. Each stage has a distinct functional responsibility and can introduce different types of failure. The evaluation framework was therefore designed to assess these stages separately as well as in combination.

The quantitative experiments evaluate the multi-agent prototype at offline, component, system, and implementation levels. The system consists of an orchestrator that interprets natural-language requests and dispatches them to specialised subagents, together with the tools and data sources used by those subagents. The evidence set covers offline properties, Routing, Database QA, Regulation/RAG, End-to-End workflow execution, Payload-independent latency, and implementation contracts.

The experimental setup was subject to two important practical constraints. First, access to the industrial partner’s internal large-language-model services was limited during the evaluation period. Second, internal vehicle data could not be used for external quantitative experimentation because of data-security and confidentiality requirements. The quantitative experiments were therefore conducted using an external language model together with programmatically generated synthetic datasets. The vehicle parameters, database records, regulatory test cases, and numerical values used in the experiments were artificial and contained no confidential industrial data.

As a result, the reported accuracy and coverage metrics should be interpreted as measurements of component behaviour on the defined and frozen evaluation benchmark rather than as direct estimates of production performance. The experiments can reveal whether the retained components behave consistently under controlled conditions, but they cannot demonstrate that identical performance would be obtained when the system is applied to real industrial vehicle data.

This limitation also motivates a distinction between two forms of evaluation evidence. The first is *dataset-dependent evidence*, obtained from benchmark experiments in which system outputs are compared with predefined expected results. The second is *implementation-level evidence*, which concerns properties of the system implementation itself, such as tool-interface contracts, deterministic-computation invariants, execution-trace completeness, and result traceability. These properties do not depend on whether the underlying vehicle records are real or synthetic and therefore provide a complementary form of evidence regarding the correctness and reproducibility of the implementation.

For the same reason, this study does not use a single end-to-end accuracy metric as the sole measure of system performance. In a multi-component execution chain, an incorrect final output may originate from different causes, including incorrect routing, faulty query generation, unsuccessful evidence retrieval, inconsistent tool interfaces, errors in deterministic computation, or failures in an external service.

Treating all such outcomes simply as “incorrect answers” would conceal their underlying causes and provide limited support for system diagnosis and improvement.

Based on these considerations, the retained evaluation is organised according to the degree of system integration at which each experiment operates. The first level is the *offline level*, which does not invoke a large language model. It is used to validate the structural integrity of the evaluation artifacts, database-interface assumptions, reference queries, and deterministic computational components.

The second level is the *component level*. The experiments at this level evaluate orchestrator routing, natural-language database querying, Regulation/RAG answering, and retrieval configuration. This separation identifies capability-specific behaviour before results are combined at system level.

Alongside these two evaluation levels, a *cross-level implementation evaluation* is used to assess compliance with declared system contracts and invariants. These checks address properties such as interface consistency, deterministic behaviour, execution preconditions, error handling, and traceability. They provide a basis for distinguishing a model-output error from a documented-interface or deterministic-calculation defect.

The resulting layered evaluation therefore serves two purposes. It measures whether the retained components satisfy their declared targets under the defined experimental conditions, and it provides evidence for explaining why failures occur when the expected behaviour is not achieved. Table 5.1 summarises the individual experiments and their roles within this evaluation framework. The experiments are described in detail in the following sections.

5.3 Evaluation Results and Analysis

This section presents the results of the evaluation experiments introduced in Section 5.2.

5.3.1 Offline-level Evaluation

Static checks confirmed that the frozen datasets satisfied the declared structure and uniqueness constraints, that the simulated database contained the required tables and fields, and that all reference SQL statements for Database QA were executable. The synthetic-database evaluation covered eight tables, cross-table consistency of vehicle identifiers, numerical ranges, and five query patterns. All declared checks passed.

The Payload component was evaluated on 60 predefined cases against an independently implemented closed-form oracle. Front axle load, rear axle load, load capacity, and limiting factor agreed in every case. Repeated calculations produced identical results for identical inputs, and all 18 cases without component changes satisfied the zero-change invariant.

Table 5.1: Overview of the evaluation experiments and their respective evaluation levels.

Evaluation level / role	Experiment	Evaluation target	Main purpose
Offline	E7: Offline Component Validity	Evaluation artifacts and deterministic components	Verify internal consistency, reproducibility, and validity of the evaluation setup.
Component	E1: Orchestrator Routing	Request routing	Evaluate whether requests are assigned to the appropriate specialist.
Component	E2: Database QA	Natural-language database querying	Evaluate query generation and database-answering performance.
Component	E3–E4: Regulation/RAG	Retrieval and regulation answering	Evaluate evidence retrieval and benchmark-specific answer behaviour.
Component	E8: RAG Ablation	Retrieval configuration	Compare the three tested chunk-size settings.
System	E5: End-to-End Workflows	Integrated tool execution	Evaluate tool selection, sequence, required output, and task success.
System	E6: Repeated Performance	Payload-independent response time	Characterise median and p95 latency by retained task type.
Cross-level / implementation	E9: Contract Compliance	Interfaces, invariants, and traceability	Identify implementation defects and verify compliance with declared system contracts.

Table 5.2: Offline validation results for deterministic components and evaluation assets.

Check	Passed	Total	Result
Dataset structure and manifest	3	3	Pass
Synthetic database validation	5	5	Pass
Payload oracle agreement	60	60	Pass
Payload determinism	60	60	Pass
Zero-change invariant	18	18	Pass
Offline contract checks	6	6	Pass

These results establish internal consistency in the evaluation assets and deterministic components. When an online task fails, non-executable reference SQL and disagreement between the Payload implementation and the test oracle can therefore be excluded as immediate explanations within the evaluated cases.

5.3.2 Component-level Evaluation

The component-level evaluation examines the functional behaviour of the main intelligent components independently from the complete end-to-end workflow.

5.3.2.1 Orchestrator Routing

The routing dataset contained 100 requests, with ten cases for each expected outcome. The ten classes comprised eight tool actions, clarification, and no-tool responses. Each request was executed independently, and the orchestrator’s first action, together with its arguments where applicable, was recorded for evaluation.

Routing performance was evaluated using accuracy and macro-F1. Accuracy measures the overall proportion of requests assigned to the correct first action, and a Wilson 95% confidence interval was reported for this proportion to quantify the uncertainty associated with the observed accuracy. Macro-F1 assigns equal weight to each routing class and therefore provides a class-balanced measure of routing performance.

The system achieved a routing accuracy of 91.0% (Wilson 95% CI: 83.77–95.19%) and a macro-F1 score of 0.9078. All 100 requests yielded valid observations, with no infrastructure errors or timeouts. For the 20 requests requiring argument verification, the generated arguments matched the reference values in all cases.

Errors were concentrated in a small number of classes. Database, Regulation, CAD, Agility, and the three RAG management actions each achieved 10/10, while Payload achieved 9/10. Clarification and no-tool requests each achieved 6/10. The orchestrator therefore identified well-specified tool tasks consistently, but was less reliable when a request lacked sufficient information or required no tool invocation.

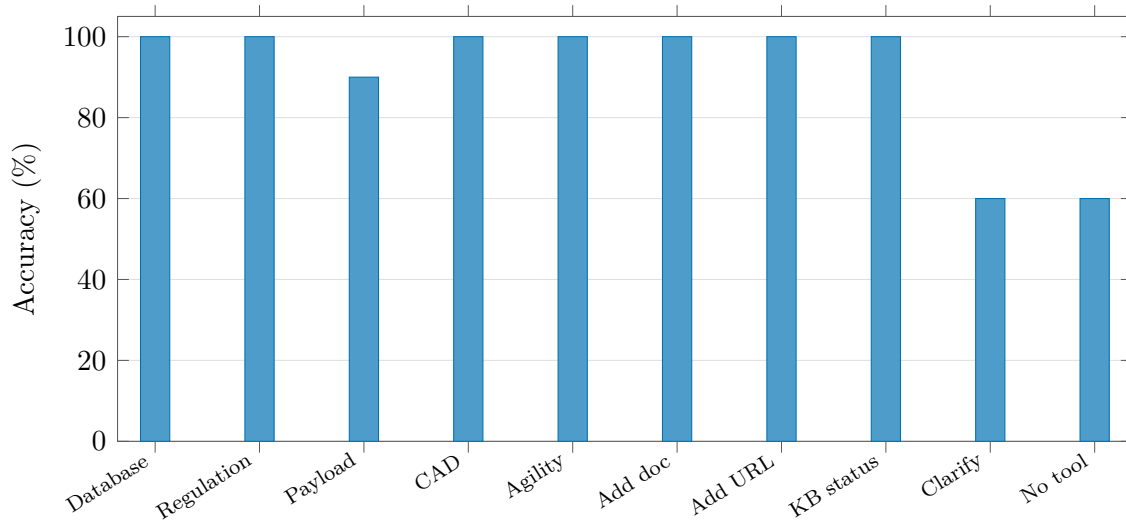


Figure 5.7: Correct first-action selection in 100 labelled routing requests, with ten prompts per expected outcome. Each bar is the percentage of prompts in that class routed to the labelled action; higher bars indicate more correct selections. Argument correctness is evaluated separately.

5.3.2.2 Database Question Answering

The Database QA dataset comprised lookup, list, aggregate, join/comparison, and negative questions, with 20 cases in each category. For each request, the system recorded the generated SQL, execution result, final answer, and latency. The SQL result was then compared with the result of the reference query.

All 100 requests produced scoreable observations. Strict execution accuracy was 54.0%, row-restriction accuracy was 65.0%, complete-value coverage was 88.0%, mean value coverage was 89.0%, and automatic answer accuracy was 79.0%. Median latency was 6.510s, and p95 latency was 10.100s.

Lookup questions achieved 25% strict execution accuracy but 100% final-answer accuracy. Several generated queries returned more columns than the corresponding reference query. They therefore failed strict result matching even though the final answer extracted the requested attribute correctly.

Negative questions achieved 35% strict execution accuracy and 0% automatic answer accuracy. The reference representation encoded an empty result as `<no rows>`, whereas the system commonly stated in natural language that the vehicle did not exist. The scorer did not treat these forms as equivalent. The result exposes a measurement problem in addition to weaknesses in query generation: the automatic criterion does not fully capture semantic correctness for empty results.

Complete-value coverage was substantially higher than strict result matching. The Database subagent often retrieved all information needed by the user, but the structure of its query output did not always match the reference SQL exactly. The main technical priorities are therefore to reduce over-broad queries, preserve the complete SQL trajectory, and introduce a machine-readable result type for empty query

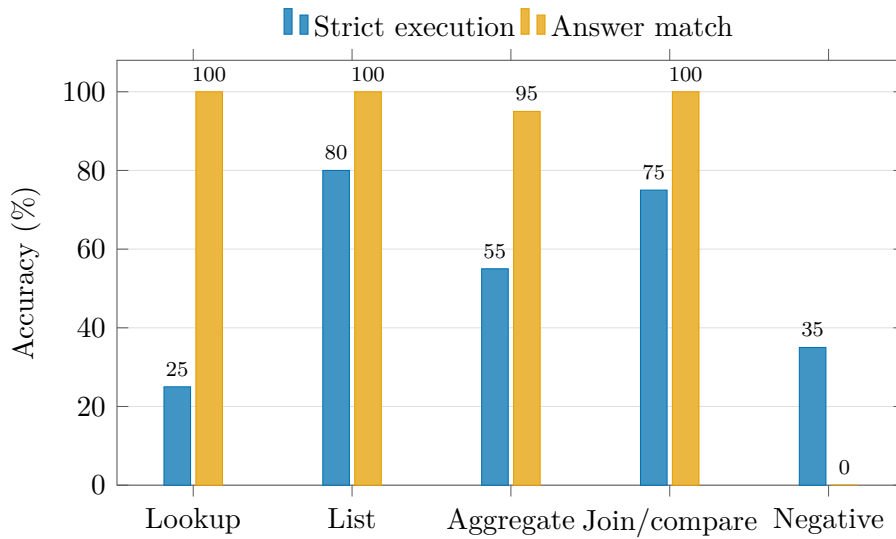


Figure 5.8: Database question answering across 100 scored requests, with 20 cases per question type. Blue bars show the percentage whose complete SQL result matched the reference result; orange bars show the percentage whose final answer matched the automatic answer target. Higher bars indicate more matches under the respective criterion; the two criteria need not agree.

results.

The Regulation/RAG and chunk-ablation results are tied to the recorded model, embedding configuration, corpus, and frozen benchmark. They characterise the evaluated configuration and are not estimates of enterprise retrieval quality.

5.3.2.3 Regulation Retrieval

The RAG component was introduced so that regulatory answers could be grounded in retrievable document content and accompanied by sources that users could inspect. This design addresses the risk of unsupported model output by making external evidence available and traceable. The experiment was not designed as a general retrieval benchmark for a large enterprise document collection.

The retrieval evaluation issued 104 queries and checked whether the expected source and annotated facts appeared among the top 3, 5, and 10 retrieved chunks. No answer-generation model was called in this experiment, allowing the vector store and embedding-based retrieval to be evaluated separately.

MRR was 0.4604, and median retrieval latency was 0.008 s. As shown in Figure 5.9, both Source Recall@ k and FactHit@ k increased with retrieval depth, while Precision@ k decreased from 0.1955 to 0.0875. This indicates a trade-off between broader evidence coverage and the inclusion of more irrelevant content.

At $k = 5$, FactHit reached 83.65%, compared with 64.42% for Source Recall. This suggests that some retrieved chunks contained the required answer facts even when they did not match the annotated source file. Duplicate regulatory content and incomplete file-level source labels may therefore cause source-based metrics to un-

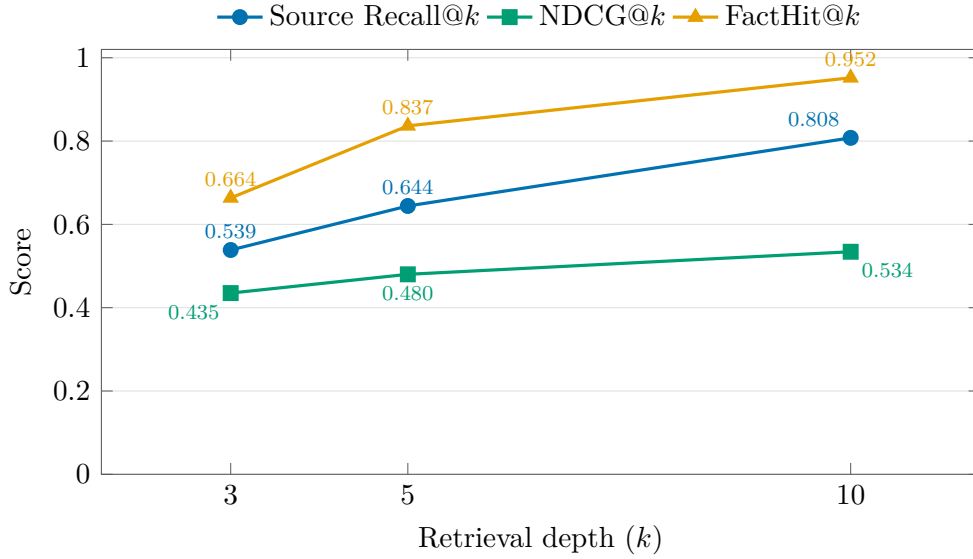


Figure 5.9: Regulation retrieval on 104 annotated queries. Each point is the mean score when the top k retrieved chunks are inspected. Source Recall@ k measures annotated-source coverage, NDCG@ k rewards earlier relevant results, and FactHit@ k records retrieval of labelled answer facts. Higher values indicate better performance under that metric; the three measures have different meanings.

derestimate the availability of relevant evidence.

5.3.2.4 Regulation Answering Strategies

Rule-only and RAG-only were evaluated on the same 104 prompts, and both completed all requests without infrastructure errors. Rule-only received the annotated region, whereas RAG-only identified the region from the question. The comparison therefore evaluates the two implemented entry points under their actual input conditions.

Table 5.3: Automatically evaluated regulation-answering results ($N = 104$ per entry).

Entry	Term match	Source match	Median latency (s)
Rule-only	47/104 (45.19%)	95/104 (91.35%)	0.0001
RAG-only	36/104 (34.62%)	65/104 (62.50%)	1.2311

The paired outcomes comprised 20 joint passes, 27 passes unique to rule-only, 16 passes unique to RAG-only, and 41 joint failures. A two-sided exact McNemar test gave $p = 0.1263$, so the automatic term-match metric did not detect a statistically significant difference. This result does not demonstrate equivalence between the strategies.

Rule-only achieved a higher automatic source-match rate and substantially lower entry latency. However, term match is a lexical measure and may reject semanti-

cally valid wording, while source match records source alignment rather than claim-level evidential support. The results should therefore be interpreted as benchmark-specific automatic outcomes, not as estimates of legal correctness or general strategy superiority.

5.3.2.5 RAG Chunk-size Ablation

The RAG ablation compared chunk sizes of 256, 512, and 1024 with overlap fixed at 64. A separate index was built for each configuration, and the same 104 queries were used in every retrieval evaluation.

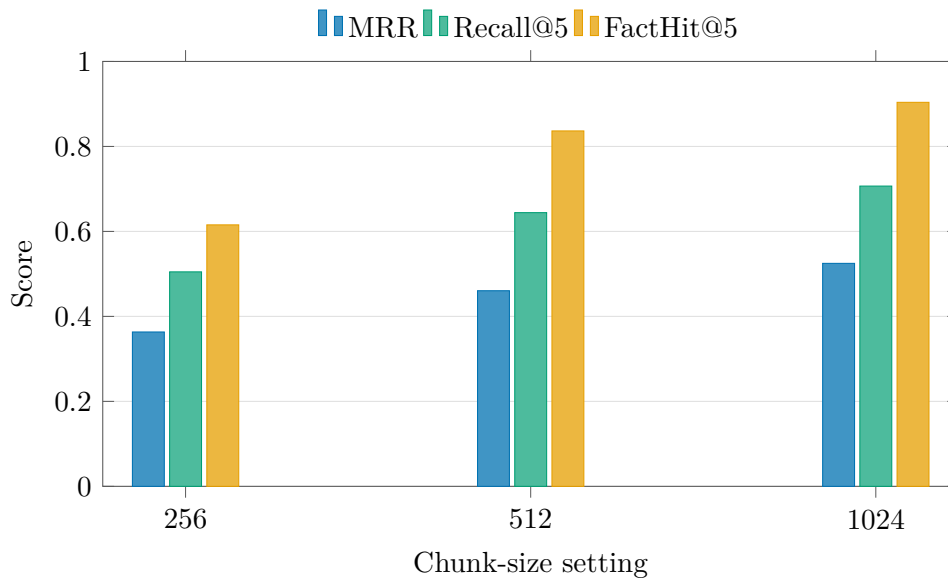


Figure 5.10: Retrieval scores for three token chunk sizes, with overlap fixed at 64 and the same 104 annotated queries evaluated for each setting. Bars show MRR (rank of the first relevant result), Source Recall@5 (annotated-source coverage in the top five), and FactHit@5 (labelled fact found in the top five). Higher scores are better on this corpus; settings are not tested on other document collections.

Within the tested range, the 1024 setting produced the highest MRR, Recall@5, and FactHit@5. The number of indexed chunks decreased from 353 at the 256 setting to 82 at the 1024 setting. The regulatory documents used in this project often required a longer context to retain a parameter name, applicable region, and numerical value in the same chunk. Smaller chunks could separate these elements and thereby reduce retrieval quality. The result supports the larger tested setting for this corpus; it does not establish 1024 as an optimal value for other document collections.

Access and confidentiality constraints limited the study to a small, curated local corpus. The restricted number and variety of documents reduce the representativeness of the experiment and make the metrics sensitive to individual files, annotations, and chunk configurations. These results primarily establish the feasibility of document ingestion, chunking, embedding, retrieval, and evidence return in the prototype. They do not estimate performance on a complete enterprise knowledge

base. Corpus size also cannot be treated as the sole explanation for the observed retrieval quality because document structure, annotation, embedding choice, and query formulation are plausible contributing factors.

5.3.3 System-level Evaluation

The system-level evaluation examines the behaviour of the integrated system when multiple components are combined into complete engineering workflows.

5.3.3.1 End-to-End Workflow Execution

The end-to-end dataset contained 50 tasks divided equally among Database, Regulation, Payload, Database-Regulation, and RAG Knowledge-base Status workflows. The evaluator checked whether the expected tools appeared, whether their sequence was correct, and whether the final answer contained the required output terms.

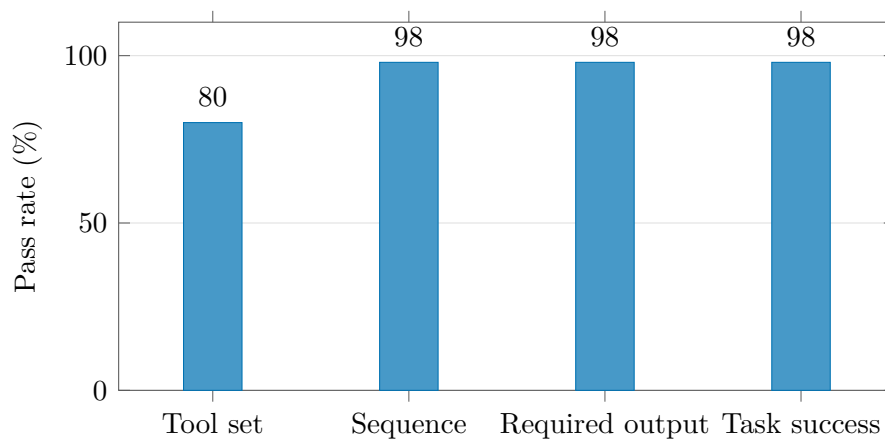


Figure 5.11: Pass rates for 50 labelled end-to-end workflows. Each bar is the percentage satisfying one criterion: exact tool set, prescribed sequence, required output, or overall task success. Higher bars indicate more passing workflows; overall success can coexist with extra tool calls.

All requests completed execution. Tool sequence, required output, and task success each passed in 49 of 50 cases, corresponding to 98.0% with a Wilson 95% confidence interval of 89.50–99.65%. Exact tool-set match was 40 of 50, or 80.0%. The only unsuccessful task failed the tool-sequence criterion.

The system therefore completed most of the specified single-tool and two-tool workflows. The difference between exact tool-set match and task success indicates that some successful tasks invoked additional tools. A Payload workflow, for example, could query the Database before executing the calculation. These additional steps did not prevent the required result from being produced, but they increased execution cost. Task success alone therefore provides an incomplete account of workflow efficiency.

5.3.3.2 Repeated Latency for Payload-independent Tasks

The repeated-performance dataset retained 140 successful observations after incomplete prompts were removed. The final thesis analysis excludes all 25 Payload observations, leaving 115 observations across 23 prompts. This exclusion is applied before interpretation, so neither the Payload median nor an aggregate influenced by Payload is reported.

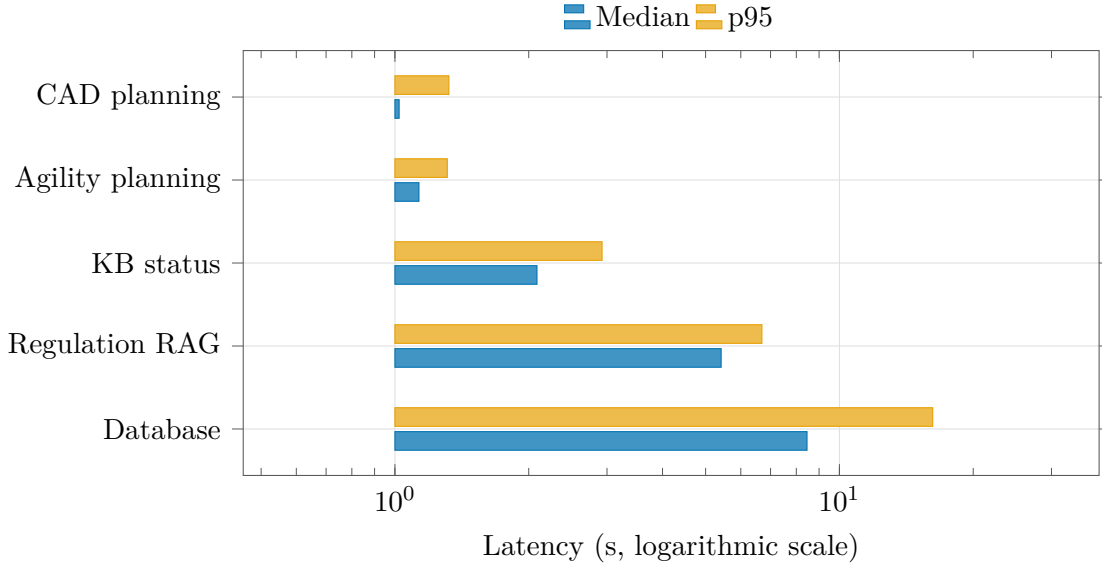


Figure 5.12: Response-time distributions for 115 retained observations across 23 prompts, excluding Payload. Each pair of bars gives median and 95th-percentile time in seconds on a logarithmic horizontal axis; shorter bars indicate faster responses. CAD and Agility measure planning only, not external software execution.

For fully executed task types, median latency was 2.090 s for Knowledge-base Status, 5.420 s for Regulation RAG, and 8.460 s for Database. Their p95 values were 2.920 s, 6.690 s, and 16.220 s, respectively. CAD and Agility planning medians were 1.020 s and 1.130 s. These values describe the recorded local setup and provider conditions. They do not estimate production latency, and the planning observations do not include complete COM execution.

5.3.4 Implementation-level Validation and Failure Diagnosis

The previous experiments identify observable performance differences and failure patterns, but do not necessarily reveal their underlying causes. Therefore, an additional implementation-level evaluation was conducted to examine whether the observed behaviour could be traced to violations of tool contracts, system invariants, or result-traceability requirements.

The evaluated Payload interface requires an axle-coded body-type label and a market value. All seven documented input cases were accepted. Across the six offline properties, Payload input acceptance, the zero-change invariant, determinism, data preconditions, typed error handling, and trace format all passed.

5.4 Summary of Evaluation Findings

Offline evaluation showed that the frozen data, reference SQL, simulated database, and Payload calculations provided an internally consistent basis for the experiments. Agreement with the independent Payload oracle in all 60 cases supports the correctness of the deterministic calculation for the evaluated input range.

At component level, routing accuracy was 91.0%. Well-specified tool tasks were handled consistently, whereas clarification and no-tool requests were the main sources of routing error. Database complete-value coverage reached 88.0%, mean value coverage reached 89.0%, and strict execution accuracy was 54.0%. Generated results therefore often contained the required values without matching the structure of the reference query.

Regulation retrieval achieved an MRR of 0.4604, Source Recall@5 of 0.6442, and FactHit@5 of 0.8365. The automatic term-match difference between rule-only and RAG-only answering was not statistically significant in the paired comparison. The 1024-token chunk setting produced the highest observed MRR among the three tested settings. Repeated latency differed by task type, but the measurements do not support a causal comparison because the workloads and execution modes differ.

At system level, tool sequence, required output, and task success each passed in 49 of 50 End-to-End cases, while exact tool-set match passed in 40 cases. The difference shows that successful workflows sometimes invoked additional tools.

All six implementation-level contract checks passed. The evaluated tool descriptions, input constraints, error types, and trace format were therefore consistent with the implementation covered by the tests.

5.5 Answers to the Research Questions

5.5.1 RQ1: Architecture and Design

RQ1 asked how a tool-orchestrated multi-agent architecture could integrate heterogeneous engineering resources for early-phase truck integration while supporting traceability and extensibility. The implemented supervisor–specialist architecture connects natural-language requests to a structured database, regulatory documents, Payload calculations, and restricted CAD and Agility operations through one orchestrator and five bounded subagents. Shared state records outputs and errors. Regulation retrieval remains an internal capability of the Regulation subagent, while knowledge-base maintenance is performed by a non-agent utility node. New capabilities can be introduced through a tool, graph node, and routing entry. Trace-format, error-typing, and determinism checks provide implementation-level evidence for this design.

5.5.2 RQ2: Prototype Capability

RQ2 asked to what extent the developed prototype correctly routed and completed its supported workflows and satisfied its declared contracts and invariants on the defined benchmark. Routing accuracy was 91.0%. Database QA achieved 54.0% strict execution accuracy, 88.0% complete-value coverage, 89.0% mean value coverage, and 79.0% automatic answer accuracy. All 60 Payload cases agreed with the independent oracle, and the six offline contract checks passed. The Regulation benchmark characterises retrieval and answering behaviour on 104 questions. In the End-to-End experiment, tool sequence, required output, and task success each passed in 49 of 50 cases, while exact tool-set match passed in 40 cases. These results support bounded component and workflow capability on the declared benchmarks, not production reliability.

5.5.3 RQ3: Performance Trade-Offs

RQ3 asked what trade-offs in accuracy, evidence faithfulness, reliability, and latency arose across the evaluated tasks and system configurations. Database QA showed a clear measurement trade-off: strict execution was 54.0%, while complete-value coverage was 88.0% and automatic answer accuracy was 79.0%. In Regulation/RAG, Source Recall@5 was 0.6442 while FactHit@5 was 0.8365, showing that file-level source matching and fact availability measure different properties. The 1024-token setting had the highest observed MRR among the three tested chunk sizes, but this is corpus-specific rather than a general optimum. Non-Payload latency also differed by task type; these workloads are not directly comparable and do not support causal efficiency claims. The thesis makes no claim about Payload latency or an overall latency value influenced by Payload.

Taken together, the retained results provide bounded feasibility evidence for the early truck-integration assistant. They show that requests can be routed to the implemented capabilities and combined in the evaluated End-to-End tasks. They also show that Database and Regulation/RAG outputs require capability-specific measures, and that deterministic Payload and interface checks can be reproduced. They do not establish broad system reliability or production performance.

6

Conclusion

Following the demonstrations and evaluation in Chapter 5, this chapter brings together the findings in relation to the original research aim and questions. It summarizes the main contributions and limitations and identifies directions for further development.

6.1 Summary of the Research

This thesis designed, implemented, and evaluated a tool-orchestrated multi-agent prototype for early-phase heavy-duty truck integration. The system accepts natural-language and structured input through a local browser interface and connects a simulated vehicle database, structured regulatory rules, local-document RAG, deterministic Payload calculations, an Agility workflow, and restricted CAD operations.

The architecture comprises one orchestrator and five ReAct subagents specialising in Database, Regulation, Payload, Agility, and CAD tasks. The LangGraph workflow also contains a RAG knowledge-base management node. This node dispatches document-ingestion and index-status tools directly and contains no independent LLM reasoning; it is therefore classified as a utility node rather than a sixth subagent. Retrieval for regulatory question answering is implemented within the Regulation subagent.

The final quantitative evidence covers offline, component, system, and implementation-level experiments. Routing accuracy was 91.0% over 100 requests, with a macro-F1 of 0.9078. Database QA achieved 54.0% strict execution accuracy, 65.0% row-restriction accuracy, 88.0% complete-value coverage, 89.0% mean value coverage, and 79.0% automatic answer accuracy. The Payload implementation agreed with an independent oracle in all 60 evaluated cases, and all six offline contract checks passed. The retained Regulation/RAG evaluation covered 104 questions, and the chunk-size ablation compared three configurations on the same queries. End-to-End task success was 49 of 50, while exact tool-set match was 40 of 50. Repeated latency results exclude every Payload observation and any aggregate influenced by those observations.

For RQ1, the study shows how a tool-orchestrated supervisor–specialist architecture can integrate heterogeneous engineering resources while supporting traceability and extensibility. The orchestrator is responsible for request interpretation, tool selection, and response composition, while each subagent performs bounded domain

Table 6.1: Summary of research questions, evidence, and supported conclusions.

RQ	Main evidence	Supported conclusion
RQ1	Resource interfaces, graph topology, trace mechanisms, and extension points	Heterogeneous resources can be integrated through one orchestrator, five bounded subagents, and one utility node with traceable execution and explicit extension points
RQ2	Routing, Database QA, End-to-End execution, oracle checks, contracts, and invariants	The prototype satisfies the tested component, workflow, and implementation properties on the declared benchmarks, although capability maturity differs across components
RQ3	Accuracy definitions, RAG ablation, and component response time	Evaluation metrics expose different aspects of capability; RAG configuration and latency results are benchmark-specific, and Payload latency is not reported

work. Shared state, trace identifiers, intermediate steps, and structured errors support runtime traceability. Tools, graph nodes, and routing entries provide the main extension points. This is a demonstrated implementation pattern for the artifact, rather than evidence of production scalability or superiority over alternative agent organisations.

For RQ2, the experiments show that the prototype correctly routed most requests and satisfied the evaluated component contracts and invariants on the defined benchmark. Routing was reliable for well-specified tool requests and weaker for clarification and no-tool cases. The Database subagent frequently retrieved all required values, although its result structure did not always match the reference query. Deterministic Payload calculations and the evaluated offline contracts were stable. In the End-to-End experiment, tool sequence, required output, and task success each passed in 49 of 50 cases. This provides benchmark-specific workflow evidence but does not establish production reliability.

For RQ3, Database QA achieved 54.0% strict execution accuracy, 88.0% complete-value coverage, and 79.0% automatic answer accuracy. Regulation retrieval achieved Source Recall@5 of 0.6442 and FactHit@5 of 0.8365. Among the three tested chunk sizes, 1024 produced the highest observed MRR. The retained non-Payload latency results range from planning-only observations to fully executed Database and Regulation tasks, so they cannot be interpreted as a controlled comparison of task efficiency. These results are bounded to the recorded benchmark and do not support claims about Payload latency or overall production efficiency.

6.2 Main Contributions

The first contribution is a working engineering artifact that connects a database, regulatory documents, calculation code, and selected engineering-software interfaces through a common natural-language workflow. The browser interface also retains structured parameter panels where precise input is preferable to free-form text.

The second contribution is a clear implementation distinction among an orchestrator, ReAct subagents, a utility node, and deterministic tools. This distinction prevents every graph node from being treated as an agent and shows where language-model reasoning is used and where direct function dispatch is sufficient.

The third contribution is runtime traceability. Each request carries a trace identifier, and worker results, tool outputs, intermediate steps, and errors are stored in shared state. This structure supports diagnosis at the routing, interface, execution, and evaluation stages rather than reducing every failure to an incorrect final answer.

The fourth contribution is an evaluation design aligned with the internal structure of the artifact. Routing, Database QA, deterministic calculation, and contract conformance use separate reference data and metrics. This design reveals component-specific weaknesses that would be hidden by a single aggregate accuracy measure.

The fifth contribution is a reproducible evaluation approach for a confidentiality-constrained industrial prototype. A simulated database and a curated local document corpus make controlled experiments possible without exposing enterprise data. The thesis also distinguishes benchmark behaviour, implementation properties, and production performance, which is necessary when the available evaluation data are not representative of the full industrial environment.

6.3 Limitations

The system remains a locally operated research prototype. It has not been deployed on an enterprise server or evaluated under production conditions such as concurrent access, identity management, network restrictions, service interruption, monitoring, and recovery. The implementation also lacks several controls required for production use, including role-based permissions, durable audit storage, approval procedures for consequential actions, and operational support arrangements. The present study therefore provides no evidence that the system can be deployed directly in a production environment.

Individual use cases were developed or exercised with the participation of one or two engineers. Their involvement provides preliminary qualitative evidence that the prototype can support the intended workflows and may improve usability and engineering efficiency for those cases. However, the participation was small-scale and was not organised as a controlled user study with standardised tasks, baseline measurements, or repeated observations. The study therefore cannot quantify the size of the improvement or generalise it to the wider population of truck-integration engineers. User acceptance, calibrated trust, long-term workflow fit, and effects on engineering quality also remain uncertain.

The RAG capability was developed with a small curated regulatory corpus because enterprise document access was restricted by confidentiality and availability. The corpus covers only a limited number of files, document types, regions, and revisions. It does not reproduce the duplication, inconsistent terminology, access rules, obsolete documents, and heterogeneous file formats found in a large internal document

repository. The Regulation/RAG experiments characterise the frozen local corpus and benchmark only. They do not establish retrieval quality, citation correctness, answer faithfulness, or hallucination reduction in an enterprise setting.

The Database evaluation used a limited simulated SQLite database. Its schema and query patterns support the prototype, but the data volume, distributions, missing values, inconsistent identifiers, historical records, complex joins, permission constraints, and concurrent workloads do not represent a large enterprise database. Data scale could affect latency, schema selection, prompt context, query generation, indexing, and the choice between direct querying and retrieval-based access. The available data were insufficient to evaluate these effects or to compare alternative Database architectures. The reported Database results are therefore benchmark-specific and cannot be transferred directly to enterprise operation.

The experiments used one chat-model deployment and one primary embedding model. Their results may depend on model behaviour, prompt wording, and provider conditions. The study also lacked a controlled monolithic-agent baseline, so it cannot establish that agent decomposition is more accurate, faster, or easier to maintain than a single-agent design using the same tools and model budget.

The End-to-End experiment is retained as benchmark-specific workflow evidence. Payload latency and the overall latency aggregate influenced by its extreme values are excluded because they do not provide a credible measure of normal system response time. The retained Regulation/RAG, chunk-ablation, and non-Payload latency observations remain subject to model, configuration, and provider-time effects. CAD and Agility were demonstrated but were not evaluated across complete industrial workflows; their engineering correctness and operational reliability remain untested.

Finally, several answer metrics relied on automatic matching, and the benchmark covers only the workflows implemented in the prototype. Paraphrases, empty results, duplicated content, and alternative valid sources can produce measurement error. The reported rates describe behaviour on the declared cases, not the probability of success across unrestricted truck-integration work.

6.4 Future Work

The first priority is a controlled enterprise pilot rather than immediate production deployment. The system should first be installed in a secured staging environment connected to approved services. This step should test authentication, role-based access, audit retention, human approval gates, monitoring, concurrent use, failure recovery, and service-level behaviour. Read-only capabilities should be validated before the system is permitted to trigger changes in engineering tools or records.

Future evaluation should expand the present small-scale engineer involvement to a broader and more diverse group of experienced truck-integration engineers. It should use tasks drawn from their actual work and measure completion rate, completion time, correction frequency, unsupported claims, and the amount of human inter-

vention. It should also collect perceived workload, usability, calibrated trust, and reasons for rejecting system outputs. A comparative or crossover design would allow the preliminary indications of improved usability and efficiency to be quantified and tested beyond the current use cases.

When governance and confidentiality conditions permit, the RAG evaluation should be repeated on an authorised enterprise document collection. The collection should include multiple file formats, document families, regions, revisions, duplicate material, and access-control metadata. Experienced engineers should construct held-out questions and annotate relevant passages, acceptable sources, and unsupported claims. Retrieval recall, citation precision and recall, answer faithfulness, and abstention quality should then be reported separately. Dense retrieval, BM25, hybrid retrieval, chunking, and metadata filtering should be compared on the same corpus and queries.

Database evaluation should use approved production-like data at several scales. If direct access to enterprise records remains impossible, possible alternatives include anonymised snapshots, a secure evaluation environment, or schema-preserving synthetic data whose similarity to the target distributions is explicitly measured. Experiments should vary row count, schema size, join complexity, missing and inconsistent values, concurrent requests, and permission boundaries. They should measure both answer quality and resource use. This would allow direct comparison of schema linking, constrained SQL generation, query repair, indexing, caching, and retrieval-based Database access.

The complete protocol should also be replicated under one fixed and fully documented configuration. The existing End-to-End, Regulation/RAG, chunk-ablation, and non-Payload latency results provide usable benchmark evidence, but replication would reduce configuration and provider-time uncertainty. Payload latency requires a new instrumented measurement. Latency should be recorded by stage so that model waiting time, Database access, retrieval, tool execution, and response composition are reported separately. This is necessary before drawing conclusions about Payload latency or overall workflow efficiency.

Finally, architectural comparisons should use a controlled monolithic-agent baseline with the same model, tools, prompts, context budget, and benchmark. CAD and Agility require separate workstation studies covering geometry validity, artifact integrity, timeout behaviour, application state, and recovery. These studies are lower priority than establishing secure deployment, representative data access, and evaluation with experienced users.

6.5 Concluding Remarks

The study demonstrates that an LLM can coordinate heterogeneous engineering resources within a bounded prototype. In the retained evidence, the system routed most benchmark requests correctly, retrieved the required Database values more often than it exactly matched the reference result structure, exposed measurable Regulation/RAG configuration effects, and satisfied the evaluated deterministic and

interface checks. Its reliability depended primarily on explicit responsibility boundaries, verifiable tools, traceable execution records, and human oversight.

The experiments also show that a successful final answer is insufficient evidence of an efficient or well-grounded workflow. Routing decisions, tool contracts, retrieval settings, scoring rules, and external software all affected system behaviour. The appropriate interpretation of the artifact is therefore an early engineering-assistance prototype with demonstrated feasibility and clearly identified development needs, rather than an autonomous engineering system ready for production deployment.

Bibliography

- [1] Liu, F., Zhou, H., Gu, B. *et al.* Application of large language models in medicine. *Nat Rev Bioeng* 3, 445–464 (2025).
- [2] Kelsall, J., Tan, X., Bergin, A. *et al.* A rapid evidence review of evaluation techniques for large language models in legal use cases: trends, gaps, and recommendations for future research. *AI & Soc* 41, 4025–4043 (2026).
- [3] José Antonio Heredia Álvaro, Javier González Barreda. An advanced retrieval-augmented generation system for manufacturing quality control. *Advanced Engineering Informatics*, Volume 64, 2025, 103007, ISSN 1474-0346.
- [4] Md Naeem Hossain, Md. Abdur Rahim, Md Mustafizur Rahman, Devarajan Ramasamy. Artificial Intelligence Revolutionising the Automotive Sector: A Comprehensive Review of Current Insights, Challenges, and Future Scope. *Computers, Materials and Continua*, Volume 82, Issue 3, 2025, Pages 3643-3692, ISSN 1546-2218.
- [5] Loureiro, G., Leaney, P.G. and Hodgson, M. (2004), A systems engineering framework for integrated automotive development. *Syst. Engin.*, 7: 153-166.
- [6] Jae Kyun Chang. *Automotive Systems Engineering*. IFAC Proceedings Volumes, Volume 41, Issue 2, 2008, Pages 12061-12064, ISSN 1474-6670, ISBN 9783902661005
- [7] OpenAI, “GPT-4 Technical Report,”arXiv preprint arXiv:2303.08774, 2023.
- [8] T. Schick et al., “Toolformer: Language Models Can Teach Themselves to Use Tools,”in *Advances in Neural Information Processing Systems*, vol. 36, 2023.
- [9] Yao et al. ”ReAct: Synergizing Reasoning and Acting in Language Models.” *International Conference on Learning Representations*, 2023.
- [10] T. Guo et al., “Large Language Model based Multi-Agents: A Survey of Progress and Challenges,”arXiv preprint arXiv:2402.01680, 2024.
- [11] Q. Wu et al., “AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation,”arXiv preprint arXiv:2308.08155, 2023.
- [12] S. Han, Q. Zhang, Y. Yao, W. Jin, Z. Xu, and C. He, “LLM Multi-Agent Systems: Challenges and Open Problems,”arXiv preprint arXiv:2402.03578, 2024.
- [13] Naihao Deng, Yulong Chen, and Yue Zhang. 2022. Recent Advances in Text-

- to-SQL: A Survey of What We Have and What We Expect. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 2166–2187, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.
- [14] Z. Hong *et al.*, "Next-Generation Database Interfaces: A Survey of LLM-Based Text-to-SQL," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 37, no. 12, pp. 7328–7345, Dec. 2025, doi: 10.1109/TKDE.2025.3609486.
- [15] P. Lewis *et al.*, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.
- [16] Patrice Béchard and Orlando Marquez Ayala. 2024. Reducing hallucination in structured outputs via Retrieval-Augmented Generation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*, pages 228–238, Mexico City, Mexico. Association for Computational Linguistics.
- [17] National Institute of Standards and Technology, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)," NIST AI 100-1, Jan. 2023.
- [18] National Institute of Standards and Technology, "AI Risk Management and Human-AI Interaction," NIST AI Risk Management Framework. [Online]. Available: <https://airc.nist.gov/airmf-resources/airmf/appendices/app-c-ai-risk-management-and-human-ai-interaction/>
- [19] Y. Lu *et al.*, "Machine Learning for Synthetic Data Generation: A Review," arXiv preprint arXiv:2302.04062, 2023.
- [20] Buggineni V, Chen C and Camelio J (2024) Enhancing manufacturing operations with synthetic data: a systematic framework for data generation, accuracy, and utility. *Front. Manuf. Technol.* 4:1320166. doi: 10.3389/fmtec.2024.1320166
- [21] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS Quarterly*, 28(1):75–105, 2004.
- [22] James Jordon, Lukasz Szpruch, Florimond Houssiau, Mirko Bottarelli, Giovanni Cherubin, Carsten Maple, Samuel N. Cohen, and Adrian Weller. Synthetic data—what, why and how? *arXiv preprint arXiv:2205.03257*, 2022.
- [23] Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. The synthetic data vault. In *2016 IEEE International Conference on Data Science and Advanced Analytics*, pages 399–410, 2016.
- [24] Ken Peffers, Tuure Tuunanen, Marcus A. Rothenberger, and Samir Chatterjee. A design science research methodology for information systems research. *Journal of Management Information Systems*, 24(3):45–77, 2007.
- [25] Model Context Protocol, *Specification*, rev. 2026-07-28. <https://modelcontextprotocol.io/specification/2026-07-28> (accessed 2026-08).

DEPARTMENT OF MECHANICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY

Gothenburg, Sweden

www.chalmers.se



CHALMERS
UNIVERSITY OF TECHNOLOGY