Degree Project in Computer Science and Engineering, specializing in Software Engineering for Distributed Systems

Second cycle, 30 credits

# Reliable Semantic Configuration of Electric Vehicle Routing Problems via LLM-based Agentic Tooling

**YILAI CHEN**

# Reliable Semantic Configuration of Electric Vehicle Routing Problems via LLM-based Agentic Tooling

YILAI CHEN

Master's Programme, Software Engineering of Distributed Systems, 120 credits
Date: June 25, 2026

Supervisor: Ki Won Sung
Examiner: Anders Västberg
School of Electrical Engineering and Computer Science
Host company: case study company; industry supervisors anonymized
Swedish title: Tillförlitlig semantisk konfiguration av elektriska fordonsruttproblem med LLM-baserade agentverktyg

© 2026 Yilai Chen

# Abstract

The transition to electric heavy-duty freight transport increases the practical complexity of daily route planning. Electric vehicle routing problem solvers can represent constraints such as battery capacity, charging availability, time windows, and energy consumption, but they often require planners to edit rigid and highly interdependent configuration files. This creates a usability barrier: operational users may understand the business intent of a change, while still finding it difficult to translate that intent into safe low-level JSON mutations.

This thesis investigates whether a large-language-model-based natural language interface can make such configuration work faster and safer when the model is constrained by deterministic tools. Following a Design Science Research approach, the work develops a prototype Decision Optimization CoPilot for electric vehicle routing configuration. The prototype combines a Streamlit chat interface, a commercial frontier large language model, a Model Context Protocol inspired tool registry, dry-run previews, rollback snapshots, and a native validation backstop. The large language model is not allowed to directly overwrite files; instead, it proposes tool calls that are executed by deterministic Python functions and checked before changes are committed.

The evaluation compares the prototype with a manual GUI interface across task scenarios derived from five complexity levels. The measured results show that the LLM CoPilot reduced average time on task from 3.8 minutes to 1.2 minutes, reduced interaction count from 35.2 manual clicks or edits to 2.5 prompts, and improved mean participant-level execution accuracy from 83.3% to 100.0%. Survey feedback also indicated higher perceived satisfaction for the CoPilot workflow. The contribution is therefore an architecture that connects semantic intent interpretation with deterministic execution, human confirmation, and validation for safety-critical configuration changes.

## Keywords

Electric Vehicle Routing Problem, Large Language Models, Natural Language Interface, Model Context Protocol, Design Science Research

# Sammanfattning

Övergången till elektriska tunga godstransporter gör den dagliga ruttplaneringen mer komplex. Optimeringslösare för elektriska fordonsruttproblem kan modellera begränsningar som batterikapacitet, laddningsmöjligheter, tidsfönster och energiförbrukning, men de kräver ofta att planerare redigerar strikta och starkt beroende konfigurationsfiler. Detta skapar ett användbarhetsproblem: operativa användare kan förstå den affärsmässiga avsikten med en ändring, men ändå ha svårt att översätta den till säkra ändringar i JSON-filer.

Detta examensarbete undersöker om ett naturligt språkgränssnitt baserat på stora språkmodeller kan göra sådant konfigurationsarbete snabbare och säkrare när modellen begränsas av deterministiska verktyg. Med Design Science Research som metod utvecklas en prototyp av en Decision Optimization CoPilot för konfiguration av elektrisk fordonsruttning. Prototypen kombinerar ett chattgränssnitt i Streamlit, en kommersiell avancerad språkmodell, ett verktygsregister inspirerat av Model Context Protocol, förhandsgranskning före körning, återställningsögonblicksbilder och ett inbyggt valideringslager. Språkmodellen får inte skriva direkt till filerna, utan föreslår verktygsanrop som körs av deterministiska Python-funktioner och kontrolleras innan ändringarna sparas.

Utvärderingen jämför prototypen med ett manuellt GUI-gränssnitt över uppgiftsscenarier från fem komplexitetsnivåer. De uppmätta resultaten visar att LLM CoPilot minskade den genomsnittliga uppgiftstiden från 3,8 minuter till 1,2 minuter, minskade antalet interaktioner från 35,2 manuella klick eller ändringar till 2,5 promptar och förbättrade den genomsnittliga exekveringsnoggrannheten på deltagarnivå från 83,3% till 100,0%. Enkätsvaren visade också högre upplevd tillfredsställelse för CoPilot-arbetsflödet. Bidraget är därför en arkitektur som kopplar semantisk tolkning av användarens avsikt till deterministisk exekvering, mänsklig bekräftelse och validering för säkerhetskritiska konfigurationsändringar.

## Nyckelord

Elektriska fordonsruttproblem, Stora språkmodeller, Naturligt språkgränssnitt, Model Context Protocol, Design Science Research

# Acknowledgments

I would like to thank my academic supervisor Ki Won Sung and examiner Anders Västberg for their guidance and feedback throughout this thesis project. I would also like to thank the industry supervisors at the case study company for their support, domain insight, and practical feedback during the development of the prototype.

## Use of AI Declaration

AI-based tools were used as supportive aids during parts of the research exploration and prototype development phases. Their use was limited to assistance such as exploring technical directions, supporting implementation work, and improving language clarity. All research choices, system design decisions, analysis, conclusions, and final writing decisions remain my own responsibility.

Stockholm, June 2026
Yilai Chen

# Contents

**1 Introduction** **1**
1.1 Introduction . . . 1
1.2 Research Background . . . 2
1.3 Problem Statement . . . 2
1.4 Research Objectives . . . 3
1.4.1 Research Purpose . . . 3
1.4.2 Research Goals . . . 3
1.5 Research Questions . . . 4
1.6 Research Methodology . . . 4
1.7 Scope and Limitations . . . 5

**2 Background and Related Work** **6**
2.1 The Electric Vehicle Routing Problem and Operational Challenges . . . 6
2.2 Large Language Models as Semantic Interfaces . . . 7
2.3 The Model Context Protocol and Agentic Tooling . . . 7
2.4 Related Work . . . 8
2.4.1 LLMs in Operations Research and Configuration Generation . . . 8
2.4.2 Reliability of Structured Output and Long-Context Challenges . . . 8
2.5 Research Gap and Motivation . . . 8

**3 Research Methodology** **10**
3.1 Theoretical Framework and System Design . . . 10
3.2 Research Process and Method . . . 10
3.3 Experimental Design . . . 11
3.4 Test Scenarios and Complexity Levels . . . 12
3.5 Data Collection and Evaluation Metrics . . . 13

3.6 Target Population and Test Subjects . . . . . 13
3.7 Reliability of the Research Design . . . . . 14
3.8 Validity of the Research Design . . . . . 14

**4 System Design and Implementation 16**
4.1 System Architecture Overview . . . . . 16
4.2 Deterministic Tool Registry . . . . . 17
4.3 Native Validation Backstop and Safety Mechanisms . . . . . 18
4.3.1 The Pure Python Validation Layer . . . . . 19
4.3.2 Pre-execution and Snapshot Rollback . . . . . 19

**5 Results and Evaluation 20**
5.1 Consolidated Performance Overview . . . . . 21
5.2 Efficiency Analysis: From Manual Editing to Semantic Intent . 21
5.3 Structural Integrity and Accuracy . . . . . 22
5.4 User Perception and Cognitive Load Analysis . . . . . 23
5.5 Reflection on Complexity Levels and Participant Backgrounds 23
5.6 Chapter Summary and Evaluation . . . . . 24

**6 Conclusion and Future Work 26**
6.1 General Conclusion . . . . . 26
6.2 Answers to the Research Questions . . . . . 26
6.3 Benefits, Ethics, and Sustainability . . . . . 27
6.4 Future Research Directions . . . . . 28
6.4.1 Automated Strategic Asset Optimization . . . . . 28
6.4.2 Dynamic Contextual Integration . . . . . 28
6.5 Final Remark . . . . . 28

**References 31**

# List of Figures

4.1 Overall architecture of the prototype . . . . . 17
4.2 Safe execution workflow for LLM-generated configuration changes . . . . . 18

# List of Tables

3.1 Experimental comparison conditions . . . . . 11

5.1 Comparative performance summary averaged across six tasks . 21

# List of acronyms and abbreviations

| | |
|---|---|
| API | Application Programming Interface |
| DSR | Design Science Research |
| EVRP | Electric Vehicle Routing Problem |
| ICE | Internal Combustion Engine |
| JSON | JavaScript Object Notation |
| LLM | Large Language Model |
| MCP | Model Context Protocol |
| NLI | Natural Language Interface |
| OR | Operations Research |
| SDG | Sustainable Development Goal |
| ToT | Time on Task |
| UI | User Interface |
| VRP | Vehicle Routing Problem |

# Chapter 1

# Introduction

## 1.1 Introduction

The global logistics industry is undergoing a transition driven by transportation electrification, as heavy-duty electric trucks (e-HDTs) enter commercial fleets [1]. This shift introduces the Electric Vehicle Routing Problem (EVRP), where planners must consider constraints such as battery capacity, charging availability, and energy consumption [2]. Although optimization solvers can model these constraints, they often require rigid JSON configurations that are difficult for operational users to maintain [3]. In large-scale cases, such as path cost matrices with tens of thousands of routing entries, manual configuration can create structural errors. Recent work suggests that Large Language Models (LLMs) can make Operations Research (OR) tools more accessible through conversational "Decision Optimization CoPilots" [4], but surveys also report unstable semantic-to-structure mapping and limited industrial deployment [5]. Long-context studies further show that LLM performance may degrade when relevant information is buried in large inputs [6], and vehicle-routing work indicates that external constraint support is important for complex routing tasks [7].

To address this gap, this thesis proposes a Natural Language Interface (NLI) integrated with a deterministic tooling layer. Rather than allowing the LLM to manipulate configuration files directly, the system uses an MCP-inspired (Model Context Protocol) tool registry and a native validation backstop to translate user intent into checked JSON updates. Following a Design Science Research (DSR) methodology [8], this work develops a functional prototype and evaluates it through task-based usability tests. The objective is to examine whether a constrained AI-assisted workflow can make

EVRP configuration more usable while preventing violations of the solver's structural rules in the evaluated tasks.

## 1.2 Research Background

The market for heavy-duty electric trucks (e-HDTs) is projected to exceed 150 billion USD by 2035 [9]. Operating these fleets requires solving the Electric Vehicle Routing Problem (EVRP), a routing optimization problem with constraints such as battery depletion, charging schedules, and disjoint vehicle operation periods. In practice, many solver workflows still rely on manual JSON or YAML configuration. For planners who are not specialists in solver data structures, maintaining consistent logic across many interrelated parameters can become a barrier to using optimization tools effectively [10, 3].

LLMs can act as semantic translators between operational intent and structured configuration, but their outputs are probabilistic. To reduce hallucination risk and keep configurations aligned with solver schemas, the model needs a controlled execution framework. In this thesis, that control is provided by an MCP-inspired tool registry and validation layer, which connect semantic interpretation to deterministic program execution [11, 7].

## 1.3 Problem Statement

The problem addressed in this thesis is the gap between solver capability and day-to-day usability. EVRP solvers can represent detailed operational constraints, but users may still need to edit complex JSON configuration files when reacting to changes such as modified time windows or temporary route restrictions [1, 12]. At the same time, unconstrained LLMs are not reliable enough to edit high-dimensional configuration files directly: long-context studies report degraded retrieval performance in large inputs [6], and routing studies show that complex VRP constraints require external support beyond simple prompting [5, 7]. The missing layer is a controlled mechanism that converts user intent into deterministic read/write operations while checking for logical conflicts before changes reach the solver [13].

## 1.4 Research Objectives

### 1.4.1 Research Purpose

The main objective of this study is to reduce the usability barrier of logistics optimization engines by moving part of the configuration workflow from manual technical editing toward semantic control. By developing a "Decision Optimization CoPilot" [4] that combines a natural language interface (NLI) with a deterministic tool abstraction layer, the study aims to support fleet planners who lack a strong background in JSON data structures or mathematical modeling. The intended outcome is a workflow that is faster to operate while still preserving validation, confirmation, and rollback for critical configuration changes.

### 1.4.2 Research Goals

To achieve the above objectives and reduce the risk of direct file manipulation by a nondeterministic model, this study sets the following three technical and operational goals:

- **Constrained Semantic Mutations:** Develop a set of atomic tools for logistics configuration. The system enforces rules such as requiring the model to clone existing objects instead of generating unsupported structures when adding vehicles or charging stations. This reduces the risk of invalid field structures [6].
- **Filter-based Bulk Matrix Updates:** For the configuration file containing tens of thousands of routing records, a controlled read/write channel is built outside the LLM context [1]. This mechanism allows non-technical users to give instructions in natural language (e.g., "add a 20% time buffer to a specific route for a certain fleet"), while the system performs batch updates through filter matching and arithmetic operations. A dry-run mechanism is used to check intent alignment and calculation results before large-scale data modifications are committed [6].
- **Deterministic Validation & Rollback:** Construct a pure Python native validation backstop independent of the compiled black-box solver. File modification operations derived from LLM intents are checked against physical and logical rules during execution and before optimizer

runs (such as checking whether time windows overlap, whether the battery attributes of EVs and fuel vehicles are compliant, and whether the self-loops on the diagonal of the matrix are zero). Once a violation is detected, the system reports the error and uses a one-click rollback snapshot mechanism to restore the previous configuration state, reducing the risk of syntax and logic violations reaching the black-box optimizer [7].

## 1.5 Research Questions

Based on these objectives, the study answers the following three research questions:

**RQ1: Architectural Design.** How to design and implement a deterministic tool abstraction layer and native validation backstop inspired by Model Context Protocol (MCP) to provide a controlled grounding mechanism for large language models (LLMs) in highly interdependent logistics configuration environments? [11, 13, 7]

**RQ2: Operational Efficacy and User Experience.** Compared with a manual GUI interface, how does the LLM-driven natural language interface affect task time, interaction count, execution accuracy, and user confidence across configuration tasks of increasing complexity?

**RQ3: Task Complexity and Boundary Conditions**. How do increasing task complexity and ambiguous operational intent affect the LLM-assisted configuration workflow, and what boundary conditions require deterministic tools, dry-run previews, validation, or user confirmation? [5, 6]

## 1.6 Research Methodology

This research uses Design Science Research (DSR). The research process includes three main steps:

- Problem Identification: Analyze the usability challenges faced by logistics operators when configuring EVRP solver inputs.
- Artifact Design & Development: Develop an MCP-inspired CoPilot prototype using a commercial frontier LLM backend, a deterministic tool registry, and a Python native validation backstop [13].

- System Evaluation: Conduct a usability comparison between the LLM application and manual operation through a manual GUI interface. The core indicators include time on task, semantic alignment accuracy, execution accuracy, and user confidence when using the dry-run system.

## 1.7 Scope and Limitations

To keep the research focused and feasible, this study sets the following technical and operational boundaries:

- Focus on Operational Planning: This system focuses on improving the agility of daily fleet scheduling (such as responding to real-time traffic failures or temporary attribute changes), and does not involve long-term strategic road network reconstruction and asset optimization [1].
- Black Box Solver: The underlying EVRP solver is treated as an unmodifiable black box. This study focuses on changes in the "configuration interface layer" and does not modify the underlying operations research algorithm.
- Hard Boundaries on AI Generation: To protect data structure consistency, operational restrictions are imposed on the AI agent. The AI component cannot create entirely new JSON fields directly; adding new objects such as vehicles must be done through a "deep-clone" of existing data.
- Language Support: The prototype was evaluated with English task prompts, but the architecture is not inherently limited to English. Since the semantic layer is provided by a multilingual LLM, future deployments could support multilingual user input after domain terminology and validation feedback are tested in the target languages.

# Chapter 2

# Background and Related Work

This chapter provides the theoretical background for the study. It introduces the operational complexity of the Electric Vehicle Routing Problem (EVRP), the role of Large Language Models (LLMs) as semantic interfaces in Operations Research (OR), and the technical ideas behind Model Context Protocols (MCPs). It then reviews related work and identifies the research gap addressed by the prototype and evaluation.

## 2.1 The Electric Vehicle Routing Problem and Operational Challenges

The electrification of heavy-duty transportation forms the industrial context for this study. Market analysis expects the electric heavy-duty truck (e-HDT) market to exceed $150 billion by 2035 [9]. This transition also introduces operational challenges for fleet scheduling. Unlike traditional internal combustion engine (ICE) routing, EVRP adds constraints such as battery consumption, charger availability, time windows, and vehicle operation periods [12, 2].

Although mathematical optimization solvers can address EVRP, they often require planners to work through highly structured configuration files. Industry survey evidence also shows that fleet electrification creates practical barriers around infrastructure, cost, range, and operational change [10]. From a cognitive load perspective, interfaces that require users to manage dense parameters, syntax, and domain rules can increase extraneous effort and make errors more likely [3]. In the EVRP setting, this problem is especially visible when planners need to update large path matrices or nested vehicle attributes

under time pressure.

## 2.2 Large Language Models as Semantic Interfaces

In recent years, large language models (LLMs) have been studied in operations research (OR) in three main roles: direct mathematical solvers, heuristic code generators, and semantic interfaces for human-computer interaction [12]. This study focuses on the third role.

Recent studies from 2024 to 2025 have explored LLMs as assistants for optimization and routing tasks. They show that LLMs can interpret natural-language task descriptions and help map them into structured representations or solver-related code [14, 15, 4]. However, LLMs remain probabilistic systems. Long-context studies show that model performance can degrade when relevant information appears in the middle of long inputs [6]. In routing configuration, this creates a practical risk: a model may understand the user's intent but still miss constraints such as non-overlapping time windows or battery limits. For this reason, direct file editing by the model is not appropriate for solver inputs; an external grounding and validation layer is needed.

## 2.3 The Model Context Protocol and Agentic Tooling

To address the conflict between probabilistic model output and rigid solver input requirements, the Model Context Protocol (MCP) offers a useful architectural reference. MCP is an open client-server protocol for connecting AI applications to external tools and data sources [11, 13].

In this thesis, MCP is used as an architectural inspiration rather than as a claim that the prototype implements the full protocol. The key design idea is that the model should not overwrite configuration files directly. Instead, it proposes calls to registered **deterministic atomic tools**, such as functions that update matrices or append array elements. This keeps semantic interpretation in the LLM while leaving file mutation, validation, and rollback to program logic. Similar MCP-based engineering discussions emphasize context control, tool selection, and privacy-preserving execution for large tool environments [16].

## 2.4 Related Work

This section reviews research at the intersection of operations research, large language model applications, and structured system configuration.

### 2.4.1 LLMs in Operations Research and Configuration Generation

Recent literature has explored how LLMs can support software and optimization workflows. The Decision Optimization CoPilot proposal argues that natural language interaction can lower the barrier to professional OR software [4]. Gulati (2025) gives an engineering case study of an LLM-based workflow configuration generator for a low-code product [17]. However, routing configuration involves deeper dependencies than many workflow schemas. Jiang et al. (2025) show that complex VRP variants benefit from external constraint retrieval and decomposition, which supports the need for architecture beyond prompt-only generation [7].

### 2.4.2 Reliability of Structured Output and Long-Context Challenges

As LLMs are used for structured outputs, reliability must be evaluated beyond JSON syntax. The STED framework argues for checking consistency and logical compliance in addition to surface validity [18]. Liu et al. (2024) also show that long-context models may fail to use information robustly when relevant content is positioned inside long inputs [6]. For EVRP configuration, these findings suggest that prompt design alone is not enough; the workflow also needs deterministic tools and validation.

## 2.5 Research Gap and Motivation

Based on the literature above, this study identifies three research gaps:

- Lack of independent filtering mechanisms for large matrices: Existing LLM-based configuration generators often rely on loading configuration content into the model context [17]. For EVRP files containing large path matrices, this approach is limited by long-context reliability issues [6]. Few studies examine controlled, filter-based bulk-update mechanisms for such files.

- Lack of evaluation of validation backstops detached from the solver runtime: Existing research emphasizes structured-output reliability and external constraint support [18, 7], but there is less work on pre-execution validation layers that check configuration rules before the black-box solver runs.
- Lack of empirical comparison of usability and confidence: There is limited empirical evidence on how an LLM semantic interface changes task time, error rate, and user confidence compared with a manual planning interface such as a manual GUI interface.

These gaps motivate the prototype and evaluation in this thesis. The work develops an MCP-inspired configuration assistant with controlled read/write channels for large matrices and a native validation layer. It then evaluates how the tool affects task time, interaction count, execution accuracy, and user confidence compared with manual JSON maintenance.

# Chapter 3

# Research Methodology

## 3.1 Theoretical Framework and System Design

This chapter describes the research methodology used in the study. It explains how the prototype was evaluated against a manual GUI interface for configuration tasks involving EVRP input data and path-matrix updates. The chapter also defines the task design, evaluation metrics, target participant background, and steps used to improve reliability and validity.

## 3.2 Research Process and Method

This study combines Design Science Research (DSR) with a task-based evaluation design. DSR is appropriate because the study builds a software artifact intended to solve an observed practical problem [8]. The research process begins by identifying the usability gap faced by logistics users when configuring operations research solvers. To address this challenge, the study develops an experimental artifact: a natural-language application that integrates an LLM with an MCP-inspired tool layer.

The methodological choices were made to match both the engineering objective and the practical constraints of the case setting. DSR was chosen because the central contribution is not only an observation about existing practice, but the design, implementation, and evaluation of a working artifact. A task-based evaluation was selected because the research questions concern observable configuration work: how long tasks take, how many interactions they require, whether the final configuration is valid, and how confident users

feel when supervising changes. A controlled comparison against the existing manual GUI workflow was therefore more suitable than an interview-only study, because it allowed the same planning requirements to be tested under both interaction modes.

The specific task design also follows from the technical risk profile of EVRP configuration. Simple attribute edits, large matrix updates, state restoration, and ambiguous operational intent create different failure modes, so the evaluation uses progressive complexity levels rather than a single generic usability task. Screen recordings were used because they make timing and interaction counts traceable after the sessions, while the post-test survey captures subjective confidence and perceived usability. Together, these choices connect the general DSR approach to the specific design decisions needed for evaluating a safety-conscious configuration interface.

After the technical design was established, the study defined a controlled comparison between two configuration modes: a traditional **manual GUI interface** and the LLM-based planning application developed for this thesis. Participants with different professional backgrounds completed the same set of JSON mutation and matrix update tasks in a controlled setting. This design makes it possible to compare task time, interaction count, execution accuracy, and user feedback across the two workflows.

## 3.3 Experimental Design

To compare the "Decision Optimization CoPilot" with the traditional manual workflow, this study uses a controlled task-based usability test.

The core mechanism of the experiment is comparative testing: participants completed the same logistics scheduling task using two different configuration modification interfaces.

Table 3.1: Experimental comparison conditions

| Condition | Description |
|---|---|
| Baseline | Using a traditional GUI interface as the manual planning tool, participants manually look up fields and enter values. |
| Experimental | Using the LLM-based prototype interface developed in this study. |

To reflect realistic planning work without exposing confidential task

details, the test questions were phrased as business requirements (for example, increased energy consumption due to icy weather). When using the LLM app, participants are not given fixed prompt templates. They are encouraged to express their intentions in their own words, which allows the evaluation to test semantic alignment under less scripted input conditions.

## 3.4 Test Scenarios and Complexity Levels

The experiment is designed around five progressive complexity levels, ranging from single-point attribute modifications to ambiguous operational preferences:

- **Level 1: Simple (Single Attribute Change).** Business Scenario: In accordance with new safety regulations, update the maximum load capacity of all electric trucks of a certain model to 25,000 kg. This aims to test the system's ability to modify a single basic parameter in an input file.
- **Level 2: Medium (Path Network Parameters).** Business Scenario: In response to a sudden traffic report, add a 20-minute travel buffer time to the route between two specific locations. This aims to test point-to-point filter updates against a large-scale path matrix JSON file.
- **Level 3: Difficult (Global Policy Update).** Business Scenario: Due to a drop in winter temperatures, a requirement is made to simultaneously increase the idle and fully loaded energy consumption of all electric vehicles globally by 10%. The goal is to test the computational accuracy and schema robustness of the system when handling global batch arithmetic operations containing tens of thousands of records.
- **Level 4: Advanced (State Management and Fault Recovery).** Business Scenario: A bridge is temporarily closed, requiring the blocking of related paths (marked as unreachable). Half an hour later, the bridge reopens, requiring the restoration of the original routing data. This scenario was used to test the system's safety mechanisms, including automatic data backup when reachable is set to false, and the system's ability to restore the original values rather than estimating missing values.
- **Level 5: Ambiguous Intent (Boundary Testing).** Business Scenario: A dispatcher issues highly ambiguous preference instructions, such

as "prioritize goods delivered in the morning." This aims to explore the technical boundaries of the current LLM toolchain when handling instructions lacking explicit mathematical mappings.

These five levels define the complexity categories used in the evaluation. The experiment contained six concrete task scenarios: one task was assigned to each of Levels 1, 2, 3, and 5, while Level 4 was represented by two linked tasks, first blocking selected paths and then restoring the original routing data after the temporary disruption was resolved.

## 3.5 Data Collection and Evaluation Metrics

To answer the research questions quantitatively and qualitatively, the system backend and test questionnaires were designed to record the following four key metrics during the test execution process:

**Time on Task (ToT):** Records the total time required from reading the business requirement to successfully implementing the change, thus measuring the operational speed of each workflow.

**Accuracy:** Uses a binary scoring system (Pass/Fail). It verifies whether the final JSON configuration conforms to the hard rules of the compiled black-box optimizer (e.g., no overlapping time windows, no illegal negative numbers) and whether the intended fields were updated correctly.

**Interaction Count:** Compares the number of manual clicks and form fills in the GUI interface with the number of prompts in LLM mode.

**Confidence Score & Qualitative Feedback:** A post-test questionnaire is used to collect users' confidence ratings (1–5) and qualitative feedback. The evaluation focuses especially on whether dry-run previews and rollback snapshots improve confidence during large matrix updates. The questionnaire itself is not reproduced as an appendix because it contained company-specific scenario wording and internal terminology. To preserve confidentiality, this thesis reports the evaluated dimensions and anonymized aggregate results instead.

## 3.6 Target Population and Test Subjects

To explore the impact of the usability gap across different technical backgrounds, the evaluation design defines three representative groups of target participants:

- **Non-expert Users**: Product managers who understand product and business goals but do not work primarily with JSON structures or operations research models.
- **Domain Experts**: Transportation planning personnel who understand logistics scheduling operations.
- **Expert Users**: Solution developers or routing-configuration power users who are familiar with the existing manual planning workflow.

## 3.7 Reliability of the Research Design

To improve reliability, the testing protocol keeps the task environment consistent. The same baseline dataset is used for both the manual GUI-interface workflow and the LLM CoPilot workflow. All participants completed the same sequence of six standardized task scenarios derived from the five complexity levels described above. By keeping the initial data, task sequence, and task difficulty consistent, the study reduces the chance that performance differences are caused by differences in the underlying problem data.

The main measurement-control measure is the use of screen recordings. Instead of relying only on self-reported completion times or live observation, operational metrics are captured asynchronously through recorded desktop interactions. The researcher can then review the recordings retrospectively. *Time on Task* and *Interaction Count* are calculated from video timestamps and visible interactions, while execution *Accuracy* is verified by reviewing the final configuration state shown during the task.

Finally, a standardized post-test survey is used to connect the objective metrics with user perception. Participants evaluate both workflows across five predefined dimensions: Intuitiveness, Usability, Efficiency, Accuracy, and Overall Satisfaction. Using the same criteria for both workflows makes the qualitative feedback easier to compare.

## 3.8 Validity of the Research Design

In addition to reliability, several validity considerations are addressed in the research design. Internal validity is supported by requiring each participant to work with the same baseline dataset and the same sequence of task scenarios in both comparison modes. This reduces the likelihood that observed differences

in task time, interaction count, or execution accuracy are caused by differences in the underlying logistics problem rather than by the interface being evaluated.

Construct validity is addressed by aligning each metric with the research questions. Time on Task and Interaction Count are used to capture operational efficiency and cognitive effort; Accuracy is used to capture whether the intended JSON fields are updated with the expected values; and qualitative feedback is used to capture perceived ease of use and confidence, especially when participants rely on dry-run previews before committing large matrix changes. Together, these measures connect the abstract concepts of usability, reliability, and confidence to observable behavior during the experiment.

External validity is expected to be more limited. The target participants are drawn from product management, solution development, and transportation planning backgrounds, and the prototype is evaluated in a controlled setting rather than in live dispatch operations. Therefore, the results should be interpreted as evidence for the feasibility and usability of the prototype in a realistic but bounded evaluation context, not as a large-scale statistical generalization to all logistics organizations. Ecological validity is strengthened by framing the tasks as business requirements rather than fixed command templates, so participants must express operational intent in their own words.

# Chapter 4

# System Design and Implementation

This chapter describes the design and implementation of the CoPilot prototype used in this thesis. It focuses on how the main components work together to support controlled local configuration editing in an EVRP planning environment.

## 4.1 System Architecture Overview

To support portability and reproducibility, the prototype was designed as a lightweight local workflow. The overall system uses three layers to separate semantic interpretation from deterministic file execution.

- **User Interface & State Layer:** The front-end is built on the Streamlit framework. This layer provides a chat-like natural language interface and a synchronized status panel. During each dialogue round, the UI parses the local configuration state and displays vehicle status and path-matrix statistics in the sidebar. This makes relevant background data visible to the user.
- **LLM Semantic Routing Layer:** This layer interprets user intent and proposes tool calls. The prototype connects to a commercial frontier LLM through a managed cloud inference service. Before each dialogue round, selected configuration context is provided to the model so that it can interpret the request against the current planning state.
- **Deterministic Execution & Interception Layer:** The model is not allowed to edit files directly. Its intent is parsed into standardized

function calls and routed through a tool registry inspired by MCP. Mutating operations are wrapped with user confirmation, rollback snapshots, and validation feedback before the resulting configuration is used by the optimizer.

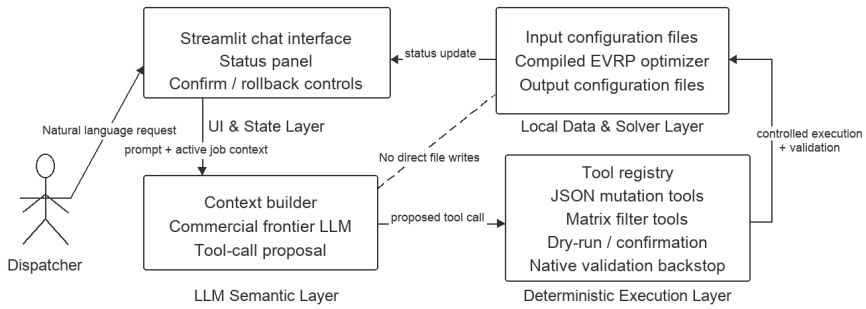


Figure 4.1: Overall architecture of the prototype

Figure 4.1 illustrates the overall architecture of the prototype, showing how natural language requests are routed through the UI layer, the LLM semantic layer, and the deterministic execution layer before reaching the local solver and configuration files.

## 4.2 Deterministic Tool Registry

As mentioned earlier, the prototype uses deterministic tools to reduce hallucination risk and avoid relying on ultra-long context processing [6]. These tools are the only permitted way for the AI component to modify solver configuration data [13]. The toolchain is divided into two execution channels:

**Constrained Mutations for Shallow Structures:** For moderate-sized input and parameter configuration files, the system provides three main tools: `patch_json_field`, `append_to_array`, and `remove_from_array`. By design, the system enforces a **cloning-based entity generation strategy**. For example, when the LLM receives an instruction to add two electric trucks to a selected fleet, it does not generate a new JSON structure from scratch. It must use existing vehicles as templates through `append_to_array`, and the system then assigns the next available identifiers. This design reduces the possibility of unsupported fields being introduced.

**Off-context filter-based matrix channels for high-dimensional matrices:** For the path matrix, which contains tens of thousands of routing records

and is too large to load directly into the LLM context, the system uses a dedicated processing channel [1]. The model only sees an abstract path-matrix update tool. It extracts matching conditions, such as fleet type or departure location, and mathematical operators, such as percentage increments through `multiply_percent`. The Python tool then performs the batch arithmetic operation locally. This design keeps the LLM focused on intent extraction rather than large-scale data computation. It also supports extensibility: new mutating tools can be added to the registry and will inherit confirmation and validation handling when marked with `is_mutating=True`.

This separation keeps the routing controller small and makes later tool additions easier to manage.

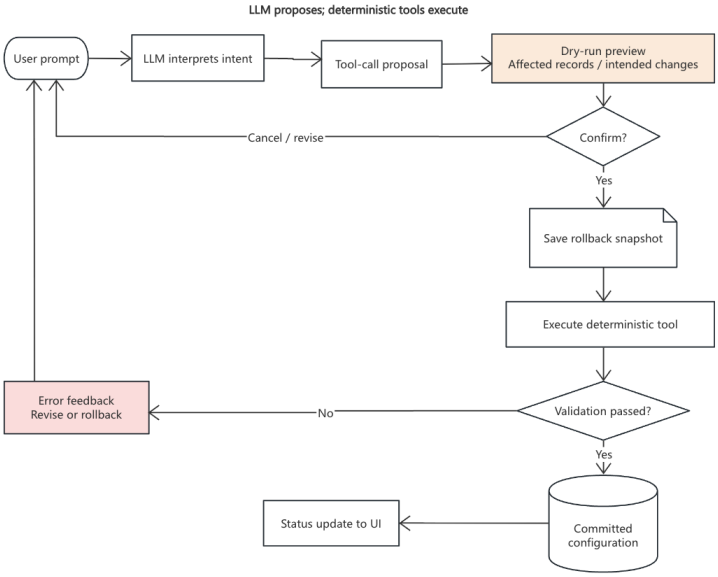


Figure 4.2: Safe execution workflow for LLM-generated configuration changes

## 4.3 Native Validation Backstop and Safety Mechanisms

In an automated configuration workflow driven by LLM, structured output may be syntactically correct (valid JSON) while still violating the underlying physical constraints of the operations research model [7]. To reduce this

risk, the system combines constrained tool execution, validation feedback, and rollback snapshots around every mutating operation.

### 4.3.1 The Pure Python Validation Layer

The underlying Electric Vehicle Routing Problem (EVRP) solver is treated as a compiled black-box engine with strict input validation logic. A configuration file with logical flaws can cause the solver run to fail. Therefore, this research implemented a native Python validation layer that mirrors the most important solver-side checks without requiring the solver runtime during routine configuration editing.

After a mutating tool executes, and again before optimizer execution, the validation layer scans the resulting data state and reports any detected inconsistencies. At the hardware attribute level, the system enforces physical isolation validation: for example, electric vehicle (EV) configurations must include complete battery module parameters, while internal combustion engine trucks (ICE) must not contain battery-related fields. In terms of time and space, the verification layer checks whether all shifts of the same vehicle are arranged in chronological order and without overlap or conflict. It also scans the high-dimensional path matrix to check that the travel time and distance of the "self-loop paths" (i.e., from a certain place to itself) on the diagonal of the matrix remain zero. When a rule violation is detected, the system formats the constraint anomalies into readable error feedback and keeps a rollback snapshot available so that the user can restore the previous configuration state [7].

### 4.3.2 Pre-execution and Snapshot Rollback

Two interaction mechanisms are included to reduce the burden of high-risk operations.

For global matrix updates that can affect tens of thousands of records, the system first performs a dry run in memory and displays the number of affected paths in the UI sidebar [1]. Disk writes are executed only after explicit user confirmation. The system also generates a rollback snapshot containing the original data before each substantial modification, giving users a practical recovery option during large-scale updates.

# Chapter 5

# Results and Evaluation

This chapter analyzes the numerical data gathered during the user evaluation sessions. The objective is to compare the LLM-based CoPilot with the manual GUI baseline in terms of task time, interaction count, execution accuracy, and user feedback.

Before presenting the results, it is important to restate the experimental context in which the data was collected. Five respondents participated in the evaluation. They were drawn from three internal user groups: product managers, solution developers, and transportation planning experts. Each participant completed the same six task scenarios using both the manual GUI workflow and the LLM CoPilot workflow. The six tasks were selected from the five complexity levels defined in Section 3.3, ranging from simple single-field edits to large-scale matrix updates and ambiguous planning intent. More detail on how participant backgrounds affected the observed results is provided in Section 5.5. To avoid exposing sensitive operational details, the result analysis reports the task categories and measured outcomes rather than the full confidential task statements.

For each task, four types of evidence were recorded. Time on Task measured the time from starting the planning activity to successful execution. Accuracy verified whether the correct JSON fields were updated with the expected values. Interaction Count measured the number of clicks or form fills in the manual method and the number of conversational prompts in the LLM method. Qualitative Feedback captured participants' subjective assessment of ease of use and confidence, especially when the LLM workflow used dry-run previews before committing changes.

## 5.1 Consolidated Performance Overview

Table 5.1 aggregates the core performance metrics collected across all test subjects. The values were calculated from screen recordings, final configuration checks, and survey forms collected during this study. The task-performance data is averaged across the six standardized task scenarios, which ranged from simple single-field updates to large matrix recalculations. The execution accuracy row reports the mean of participant-level task success rates, and the survey score in the final row reports overall satisfaction rather than a single usability sub-dimension.

Table 5.1: Comparative performance summary averaged across six tasks

| **Metric** | **Baseline: Manual GUI** | **Experimental: LLM CoPilot** | **Improvement / Variance** |
|---|---|---|---|
| Average Time on Task (ToT) | 3.8 minutes | 1.2 minutes | -68.4% |
| Interaction Count | 35.2 clicks/edits | 2.5 prompts | -92.9% |
| Execution Accuracy | 83.3% mean participant success rate | 100.0% mean participant success rate | +16.7% |
| Overall Satisfaction Score (1–5) | 2.6 | 4.8 | +84.6% |

The baseline accuracy of the manual GUI remained reasonably high, but maintaining this accuracy required more time as scenarios became more demanding.

## 5.2 Efficiency Analysis: From Manual Editing to Semantic Intent

The clearest efficiency pattern was the increase in manual effort required for high-dimensional updates.

In Level 1 and Level 2 tasks (localized parameter changes, such as modifying a specific vehicle's capacity), expert users maintained a high baseline efficiency. Using the manual GUI, operators typically navigated to the correct JSON objects and executed the changes within 1 to 2 minutes. The LLM CoPilot handled these same tasks in 0.5 to 1 minute. For these lower-complexity operations, the time advantage of the LLM was measurable but not decisive, as human familiarity with the existing tool remained competitive.

However, as tasks progressed to Level 3 and 4 (global matrix updates), the manual workflow became more time-consuming. When implementing a temporary energy consumption buffer across a fleet matrix, operators using the manual GUI took between 3 and 4 minutes per task. Screen recording analysis showed that users spent much of this time on navigation and verification, including scrolling and cross-referencing to ensure no nested records were missed.

In contrast, the LLM CoPilot allowed the operator to avoid direct navigation of the data structure. By issuing a semantic command, the user could trigger a predefined tool that executed the batch change after preview and confirmation. Across the complex scenarios, the LLM workflow maintained an average ToT of just under 2 minutes. This shift from manual data manipulation to intent supervision accounts for the overall 68.4% reduction in average ToT.

## 5.3 Structural Integrity and Accuracy

In industrial logistics, configuration speed is only useful if structural safety is preserved. An illegal parameter, such as an overlapping time window or a negative distance value, can cause the underlying compiled EVRP solver run to fail.

The video analysis of the manual GUI sessions revealed a mean participant-level success rate of 83.3% across the six standardized scenarios. The observed failures were not primarily caused by a misunderstanding of the business requirement, but by structural configuration errors.

In contrast, the LLM CoPilot achieved a mean participant-level execution accuracy of 100% across the six standardized scenarios, with zero structural or syntactic violations in the committed configurations. This result is linked to the system's "Native Validation Backstop" and rollback-aware execution flow. The screen recordings also showed that ambiguous Level 5 prompts could still produce uncertain intermediate tool choices, but the dry-run preview and validation feedback made these risks visible before users committed large-scale changes. This result suggests that the deterministic tool layer does not remove all ambiguity from natural language, but it reduces the chance that such ambiguity becomes an invalid solver input.

## 5.4 User Perception and Cognitive Load Analysis

The standardized five-dimension post-test survey was used to compare how participants perceived each workflow. In this evaluation, cognitive effort was considered through two observable indicators: the density of manual interactions and the user-reported Intuitiveness and Usability scores.

The data shows a clear contrast in how operators navigated the data structures. When using the manual GUI, the system required an average of 35.2 manual interactions (clicks, scrolling, and cell edits) per task. Video analysis showed that users often had to scan dense interface rows, locate the correct field, and type the updated value while checking adjacent parameters. This field-level maintenance increased the need for careful attention. The manual baseline received lower scores in both Intuitiveness (2.1 / 5.0) and Usability (2.5 / 5.0), reflecting the friction users experienced when translating business goals, such as adding a weather buffer, into low-level configuration coordinates.

The LLM CoPilot changed this interaction model. By shifting the input method from tabular cell targeting to natural language commands, the physical Interaction Count decreased by 92.9% to an average of 2.5 prompts. Operators expressed their intent and inspected the resulting preview instead of manually locating each affected record. This reduction in interaction effort is reflected in the survey data: the CoPilot achieved an Intuitiveness score of 4.8 / 5.0 and a Usability score of 4.7 / 5.0.

## 5.5 Reflection on Complexity Levels and Participant Backgrounds

The five complexity levels were reflected most clearly in how the performance gap changed as task difficulty increased. For Level 1 and Level 2 tasks, the manual baseline remained relatively competitive because the changes were localized and experienced users could rely on familiarity with the existing tool. In contrast, Level 3 and Level 4 tasks required larger-scale matrix or state-management operations, where manual navigation, verification, and rollback handling became more time-consuming and error-prone. The LLM CoPilot showed its largest advantage in these higher-complexity tasks because the user could express the planning intent semantically while the deterministic tools

performed the structured update.

Level 5 highlighted a different type of advantage. For users who were not familiar with optimizer parameters or the detailed structure of the input and output files, ambiguous instructions were difficult to translate into safe manual edits. In these cases, the LLM interface helped users explore the meaning of parameters, understand the current input and output state, and reason about possible relationships between configuration changes and solver outcomes. It could also turn vague operational intent into a more structured and executable form before routing it into constrained tool calls. The value of the CoPilot in Level 5 was therefore not only speed, but also its ability to connect informal business language with safe execution functions.

The participant backgrounds were used to test whether the interface could support different user perspectives, while the main result presentation remains focused on aggregate outcomes across all users. The observations were consistent with the intended evaluation design: product managers and other non-expert users benefited most from not needing to understand the underlying JSON structure; transportation planning experts valued the ability to express operational requirements directly; and solution developers or VRP power users remained efficient in simple manual tasks. In comparison, the LLM CoPilot provided a more direct and efficient path for larger-scale updates by reducing the need for manual navigation and low-level configuration editing. Therefore, the results suggest that the CoPilot is most useful when task complexity increases or when users have domain knowledge but limited familiarity with solver configuration syntax.

## 5.6 Chapter Summary and Evaluation

This chapter evaluated the LLM-based Decision Optimization CoPilot against a traditional manual GUI baseline. The data came from asynchronous screen recordings and standardized subjective surveys across six scenarios derived from five complexity levels.

The findings show that expert operators can maintain reasonable accuracy using manual GUI tools for simple localized updates, but the workflow becomes less efficient as routing complexity increases. The manual approach required more time (averaging 3.8 minutes overall and peaking in Level 4/5 tasks) and more interactions (35.2 clicks/edits per task). This manual effort is the main usability barrier addressed by the prototype.

By introducing an MCP-inspired tool registry and a native validation backstop, the CoPilot prototype reduced the effort required for these tasks. The

quantitative results show that abstracting the solver's JSON schema behind a natural language interface reduced execution time by 68.4% and avoided the structural failures observed in the manual baseline during the evaluated tasks. The survey metrics also indicate that this architecture made the configuration tasks easier to perform. In this workflow, the dispatcher acts less as a low-level data editor and more as an intent reviewer supported by deterministic tools.

# Chapter 6

# Conclusion and Future Work

## 6.1 General Conclusion

The electrification of freight logistics introduces new routing challenges. While modern operations research solvers can model the Electric Vehicle Routing Problem (EVRP), their practical use can be limited by configuration complexity. This thesis addresses usability and structural-safety issues found in manual configuration workflows such as a manual GUI interface.

Using Design Science Research (DSR), this study develops and evaluates a "Decision Optimization CoPilot" for EVRP configuration. The prototype shows how LLMs can be constrained by an MCP-inspired tool registry and a Python validation backstop. By shifting the interaction model from tabular data maintenance to semantic intent supervision, the evaluation indicates that the system can make configuration tasks faster and easier to control without allowing the LLM to directly overwrite solver inputs.

## 6.2 Answers to the Research Questions

**RQ1: Architectural Design.** The thesis answers RQ1 by proposing a layered architecture in which the LLM performs semantic interpretation, while deterministic tools perform all file mutations. The tool registry, dry-run preview, user confirmation, validation backstop, and rollback snapshots provide the controlled grounding mechanism needed for interdependent EVRP configuration files.

**RQ2: Operational Efficacy and User Experience.** The evaluation with five respondents indicates that the CoPilot improved the measured workflow in the tested scenarios. Average task time decreased from 3.8 minutes to 1.2

minutes, interaction count decreased from 35.2 clicks or edits to 2.5 prompts, mean participant-level execution accuracy increased from 83.3% to 100.0%, and overall satisfaction increased from 2.6 to 4.8 on a five-point scale.

**RQ3: Task Complexity and Boundary Conditions.** The results show that the CoPilot is most useful when tasks involve bulk updates, state restoration, or users who understand the business intent but not the configuration structure. Ambiguous instructions remain a boundary condition: the system should not treat vague preferences as direct executable changes without deterministic tools, previews, validation, and user confirmation.

## 6.3 Benefits, Ethics, and Sustainability

The main practical benefit of the prototype is that it reduces the effort required to operate complex EVRP optimization tools in the evaluated configuration tasks. In freight operations, planners often need to respond quickly to disruptions, changing delivery requirements, vehicle constraints, and charging limitations. By translating operational intent into controlled configuration updates, the CoPilot reduces manual data-entry work and helps users focus on planning decisions rather than JSON syntax. This is especially relevant for electric heavy-duty transport, where battery capacity, charger availability, charging time, range constraints, and operational schedules make planning more complex than in conventional diesel-based routing.

From an ethical perspective, the system is designed as a human-supervised decision-support tool rather than an autonomous decision-maker. The LLM does not directly overwrite files without user confirmation, and high-risk operations are supported by dry-run previews, validation feedback, and rollback snapshots. These design choices preserve human accountability and reduce the risk that a probabilistic model silently introduces invalid or harmful configuration changes. The system also raises data governance considerations: logistics configuration files may contain commercially sensitive operational data, so deployment should be limited to approved environments with appropriate access control, data retention policies, and clear responsibility for final planning decisions.

From a sustainability perspective, the work supports the broader transition toward electric freight operations. Heavy-duty transport is a major contributor to freight-related emissions, and electrification can help reduce local pollution and support decarbonization when combined with suitable energy systems and operational planning [1]. However, electrification also introduces new planning burdens, including charging infrastructure constraints, range

limitations, and trade-offs between vehicle utilization and battery capacity [10]. A more usable planning interface may help organizations test and manage these constraints more effectively. At the same time, this thesis does not claim that the interface alone guarantees emission reductions; sustainability benefits depend on how the optimized plans are used, the electricity mix, asset utilization, and the broader logistics strategy.

## 6.4 Future Research Directions

The current prototype mainly reacts to human instructions for existing scheduling needs. Future research could extend it toward more proactive planning support in the following areas:

### 6.4.1 Automated Strategic Asset Optimization

Future iterations of the CoPilot could move beyond daily routing adjustments to support long-term capital expenditure analysis. An automated experimentation loop could generate and test "what-if" fleet configurations, such as comparing the lifecycle ROI of 200kWh and 300kWh battery capacities under seasonal constraints. The system could run these alternatives through the EVRP solver, compare the resulting KPIs, and support hardware purchasing decisions with data rather than manual scenario editing.

### 6.4.2 Dynamic Contextual Integration

Currently, the system relies on human dispatchers to input external disruptions, such as adding a buffer due to snow. Later research could integrate the CoPilot with real-time telemetry APIs, live traffic feeds, or weather data. This would allow the system to detect external changes, draft candidate matrix updates, and present the proposed mitigation to the dispatcher for review.

## 6.5 Final Remark

As heavy-duty electric transport grows, usability will remain an important factor in whether optimization tools can be used in daily operations. This thesis contributes a deterministic framework for integrating generative AI into routing configuration while preserving human review and programmatic validation. The evaluation results suggest a clear usability benefit in the tested

scenarios, while future work should examine the same architecture with larger participant samples and live operational integrations.

# References

[1] "Master thesis proposal 2026," 2025, internal document, case study company.

[2] M. Meraliyev, C. Turan, S. Kadyrov, and U. Sadyk, "A comprehensive survey of methods and challenges of vehicle routing problem with uncertainties," *Mathematics*, vol. 13, no. 23, p. 3782, 2025. doi: 10.3390/math13233782. [Online]. Available: https://www.mdpi.com/2227-7390/13/23/3782

[3] P. Salyha and O. Sytnyk, "Interface design based on cognitive load theory," *Demiurge: Ideas, Technologies, Perspectives of Design*, vol. 8, no. 2, pp. 241–258, 2025. doi: 10.31866/2617-7951.8.2.2025.347381. [Online]. Available: https://demiurge.knukim.edu.ua/article/view/347381

[4] S. Wasserkrug, L. Boussioux, D. den Hertog, F. Mirzazadeh, I. Birbil, J. Kurtz, and D. Maragno, "From large language models and optimization to decision optimization copilot: A research manifesto," 2024. [Online]. Available: https://arxiv.org/abs/2402.16269

[5] Y. Wang and K. Li, "Large language models in operations research: Methods, applications, and challenges," 2025. [Online]. Available: https://arxiv.org/abs/2509.18180

[6] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang, "Lost in the middle: How language models use long contexts," *Transactions of the Association for Computational Linguistics*, vol. 12, pp. 157–173, 2024. doi: 10.1162/tacl_a_00638. [Online]. Available: https://aclanthology.org/2024.tacl-1.9/

[7] X. Jiang, Y. Wu, C. Zhang, and Y. Zhang, "Droc: Elevating large language models for complex vehicle routing via decomposed

retrieval of constraints," in *International Conference on Learning Representations*, 2025. [Online]. Available: https://proceedings.iclr.cc/paper_files/paper/2025/hash/73d6c3e4b214deebbbf8256e26d2cf45-Abstract-Conference.html

[8] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004. doi: 10.2307/25148625. [Online]. Available: https://doi.org/10.2307/25148625

[9] Global Market Insights, "Electric heavy duty trucks market size & share 2026–2035," 2026. [Online]. Available: https://www.gminsights.com/industry-analysis/electric-heavy-duty-trucks-market

[10] Qmerit, "Ev fleet conversion survey: Real-world insights from fleet professionals," 2025. [Online]. Available: https://qmerit.com/blog/2025-industry-insights-ev-fleet-conversion-survey/

[11] X. Hou, Y. Zhao, S. Wang, and H. Wang, "Model context protocol (mcp): Landscape, security threats, and future research directions," 2025. [Online]. Available: https://arxiv.org/abs/2503.23278

[12] U. Zafar and S. Bayhan, "Using large language models to solve the electric vehicle routing problem with advanced prompting techniques," in *2025 IEEE 19th International Conference on Compatibility, Power Electronics and Power Engineering (CPE-POWERENG)*, 2025. doi: 10.1109/CPE-POWERENG63314.2025.11027311. [Online]. Available: https://elmi.hbku.edu.qa/en/publications/using-large-language-models-to-solve-the-electric-vehicle-routing

[13] Model Context Protocol, "Architecture overview," 2025. [Online]. Available: https://modelcontextprotocol.io/docs/learn/architecture

[14] Z. Huang, G. Shi, and G. S. Sukhatme, "Can large language models solve robot routing?" 2024. [Online]. Available: https://arxiv.org/abs/2403.10795

[15] K. Li, F. Liu, Z. Wang, X. Tong, X. Han, M. Yuan, and Q. Zhang, "Ars: Automatic routing solver with large language models," 2025. [Online]. Available: https://arxiv.org/abs/2502.15359

[16] A. Jones and C. Kelly, “Code execution with mcp: Building more efficient agents,” 2025. [Online]. Available: https://www.anthropic.com/engineering/code-execution-with-mcp

[17] S. Gulati, “Case study: Building a llm-based workflow configuration generator for a low-code product,” Jan. 2025. [Online]. Available: https://shekhargulati.com/2025/01/26/case-study-building-a-llm-based-workflow-configuration-generator-for-a-low-code-product/

[18] G. Wang, J. Yu, X. Zhang, D. Jiang, Y. Song, T. Deb, X. Liu, and P. He, “Sted and consistency scoring: A framework for evaluating llm structured output reliability,” 2025. [Online]. Available: https://www.amazon.science/publications/sted-and-consistency-scoring-a-framework-for-evaluating-llm-structured-output-reliability

TRITA-EECS-EX-2026:405
Stockholm, Sweden 2026

www.kth.se