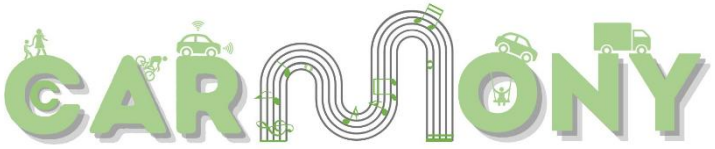

Agent-based simulation controllers and simulation platform – Initial Version

Deliverable Number	D3.3
Deliverable Type	OTHER
Dissemination Level	PU - Public
Deliverable Responsible	Kostas Cheliotis (FRON)
Document Version & Status	V1.0 | Final

Project Acronym	CARMONY
Project Title	Coordinated Action for Responsive Mixed Orchestration and Network Yield
Grant Agreement Number	101202858
Project Coordinator	Virtual Vehicle Research GmbH
Project Website	www.carmony-project.eu

The document is submitted to the European Commission for review but has not been accepted so far. As soon as the final version of this document is available, this document should be deleted.

Funded by the European Union

CARMONY has received funding from the European Union's Horizon Europe research and innovation programme under Grant Agreement No. 101202858.

Author(s)

Name	Organisation
Kostas Cheliotis	FRON
Felix Tischer	ViF
Gerhard Benedikt Weiß	ViF

Reviewers

Name	Organisation	Date
Ana Martínez Roselló	ETRA	2026-07-09
Juan Carlos Cantos	ETRA	2026-07-09
Behnood Dianat	UNILUX	2026-07-21

Change History

Version	Date	Name/Organisation	Description
0.1	05/05/2026	Kostas Cheliotis/FRON	Initial Table of Contents
0.2	27/05/2026	Kostas Cheliotis/FRON	Finalized Table of Contents
0.3	19/06/2026	Felix Tischer/ViF, Gerhard Benedikt Weiß/ViF	Sections 3, 5
0.4	02/07/2026	Kostas Cheliotis/FRON	Sections 2, 6.1
0.5	07/07/2026	Kostas Cheliotis/FRON	Sections 1, 4, 6.2, 7
0.9	07/07/2026	Kostas Cheliotis/FRON	Proofreading
1.0	26/07/2026	Felix Tischer/ViF, Gerhard Benedikt Weiß/ViF, Kostas Cheliotis/FRON	Added Section 3.1, Addressed reviewers' comments, final proofreading

Table of Contents

List of Figures

List of Tables

1. Executive Summary

This deliverable (D3.3) presents the initial version of the Simulation System for the CARMONY project. It describes the key components that comprise the overall Simulation System and highlights its applicability via a video demonstration of the Simulation System in action for one of the CARMONY Use Cases.

The Simulation System is a key part of the CARMONY Orchestrator architecture, and as such its design has been informed significantly by the needs of the rest of the project's technical components. In order to support the simulation needs of the project, the overall Simulation System has been designed with two core features in mind: First, it provides access to a robust traffic simulation engine that is capable of handling scenarios of varying spatial and temporal scales and fidelities. And second, it is designed in an interoperable approach, allowing it to be seamlessly integrated within the rest of the CARMONY system.

The various components that support the two core features are presented in detail, including the Simulation Framework's technical layers and processes that interface with the traffic simulation engine, the Interoperability Framework's architecture that supports its interfacing with the rest of the CARMONY systems, and the many Agent-Based Network Management Controllers that capture and emulate relevant processes and systems, enriching the simulation engine and enabling more complex orchestration scenarios to be examined.

While still at an early stage of development, the capabilities of the Simulation System are further showcased via a demonstration that presents the Simulation System in action. The video demonstration captures the full process of initializing and running a traffic simulation within CARMONY, with one of the project's Use Cases being used as the template for the simulated scenario.

Through the detailed description of the main components comprising the Simulation System and the demonstration of the System in action, its capabilities and readiness at this stage are established, and the foundation is provided that will support the complex simulation scenarios that sit at the core of the CARMONY Orchestration System.

Keywords: Simulation Framework, Agent-Based Controllers, Interoperability Framework, Traffic Simulation

2. Introduction

2.1 Objectives of this Deliverable

The main aim of this deliverable is to document and present the work so far in the CARMONY project that relates to the development and delivery of an Agent-Based Simulation Platform, an integrated simulation framework for the modelling of network management orchestration solutions. As such it presents work performed under Tasks 3.4: Network management simulation controllers and 3.5: Setup & testing simulation framework.

In order to support this core aim, we introduce two secondary objectives that this deliverable will achieve: First, we will present in detail the various systems designed and developed in support of the simulation framework. And second, we will demonstrate how the simulation system will perform during operational status, to provide a clear picture of the capabilities of the simulation system.

This deliverable consists of two parts that work complementary to each other, in order to best achieve its objectives. The first part is this report that contains all of the technical documentation, presentations, and explanations of the systems in question. The second part is a video file that demonstrates the simulation system during operation. These two parts together achieve the core objective of documenting the initial version of the CARMONY Agent-Based Simulation Platform: The video showcases the described system during operation, and the report highlights and explains the processes observed in the video.

2.2 Links with WP3 tasks and objectives

2.2.1 Links with WP3 tasks

This report continues from work undertaken in Task 3.1 and presented previously in deliverable D3.1, which outlined the architecture of the whole CARMONY system. Where Task 3.1 identified the need for simulation processes in the architecture, this deliverable will describe how these simulations will be set up and operate. Furthermore, the technical development work in CARMONY is iterative: the initial version of the architecture presented in D3.1 has informed the initial implementation of the simulation system presented in this report, which in turn will highlight issues and adaptations that will be included in future work of Task 3.1, providing a feedback loop.

This report also has strong links with work undertaken in Tasks 3.2 and 3.3. More specifically, Task 3.3 focusses on mitigation measures to handle traffic disruptions and incidents and therefore has a clear link with the work presented in this document, as these measures will be evaluated in simulation environments first and foremost, to examine their effectiveness and identify their potential. Furthermore, Task 3.2 focusses on analysing traffic conditions to detect and forecast issues and degradation in the traffic network. While the systems developed in Task 3.2 are primarily calibrated on real-world data, it is often the case that specific real-world data can be sparse or not include vital information. In such cases the simulation system can generate synthetic data that is highly realistic, which can then be used by the traffic analysis tools of Task 3.2 to fine-tune them further.

2.2.2 WP3 objectives

This deliverable presents the initial version and progress so far on the work related to simulations in the CARMONY project, which is all part of WP3. As such it directly addresses a number of the WP3 objectives, as identified in the Grant Agreement.

First, one of the objectives of WP3 is to define interoperability interfaces that enable the various tools and components that constitute the CARMONY Orchestrator to cooperate with one another. This is directly fulfilled as part of Task 3.4 that develops the Interoperability Framework, a system that allows the various systems to interconnect with one another in a seamless manner by providing an interface layer to handle all inter-component communication. This interface removes the need to hard-code communication between the various modules, reducing development time and allowing for quicker adaptation and substitution of systems. The Interoperability Framework is discussed in more detail later on in Section 4.3.

Another objective of WP3 is to create an agent-based simulation framework to support the various simulation needs of the project. This is addressed in two ways: Task 3.4 develops a suite of agent-based simulation controllers that emulate the behaviour of various elements and entities relevant to the environments to be simulated in CARMONY, therefore allowing for synthetic replicas to be used in simulations in place of real-world processes (discussed in more detail in Section 4.2). And secondly, Task 3.5 develops the traffic simulation system that handles the simulation of vehicle movements on the road (presented in Chapter 5), done via an agent-based microsimulation engine.

Finally, WP3 is tasked with enabling the evaluation of complex orchestration scenarios. This objective is addressed first and foremost throughout this deliverable, as its core purpose is to present the CARMONY simulation system that will be used for testing the various traffic optimizations generated within the CARMONY project. Additionally, the objective is further addressed in Chapter 6, where we present a demonstration of the simulation system in action, applied to a scenario replicating one of the core Use Cases of the CARMONY project.

2.3 Structure and content of this Deliverable

The rest of this document is structured as follows:

This section provided an overall introduction, highlighting the objectives of this deliverable, how it links with the rest of the WP3 tasks and how it addresses the WP3 objectives, and we also provide here an overview of the rest of the document.

The following section (Section 3) will present an overview of the whole simulation system used in CARMONY. It will outline the main components, identify their core functionalities, and describe how they relate to and connect with one another. Additionally, it will present the different modes of simulation required in CARMONY, and how they address the needs of the CARMONY architecture.

Section 4 discusses the Network management simulation controllers in detail, which contains work and components undertaken in T3.4. This includes modules that simulate the functionalities of the various systems and entities needed within the simulation system, as well as an interoperability framework that allows all of the different elements to combine and cooperate with one another.

In the following section (Section 5) we present the simulation framework in detail, focussing on two main components: First, the actual simulation engine used to simulate traffic on road

networks, and second the systems built around it that enable its connection with the rest of the components presented here, and therefore the rest of the CARMONY system in general.

The sections up until (and including) Section 5 will have presented all of the technical developments so far regarding the CARMONY Simulation System. The following section (Section 6) will present a sample workflow that demonstrates the simulation system's functionalities in action. It highlights the assumptions under which the demonstration operates, and outlines the processes that take place.

Finally, Section 7 concludes the main body of this report. It offers a summary of the document, highlights its main achievements, notes the main challenges and open issues of the CARMONY simulation system, and outlines the next steps that this work will focus on.

3. Simulation system overview

This chapter provides an overview of the simulation system developed within CARMONY to support the evaluation of traffic orchestration concepts and network management strategies. The system combines agent-based simulation controllers, traffic simulation tools, CARMONY (orchestration) services, and an interoperability framework into a modular environment for the assessment of the project use cases.

As illustrated in Figure 1, the simulation system consists of four main elements:

- **Simulation controllers**, which represent the network management entities and traffic participants. These include road network and infrastructure controllers, public transport controllers, parking management systems, fleet operators, and individual vehicle and traveller agents.
- **CARMONY services**, which implement orchestration functionalities and use-case-specific services, such as route guidance, booking services, or traffic regulation strategies.
- **Simulation framework**, which provides the traffic simulation environment used to evaluate orchestration actions. Depending on the use case and evaluation objective, different simulation modes and fidelity levels can be employed.
- **Interoperability framework**, which acts as the central communication layer between all components and enables the exchange of simulation data, control actions, and orchestration decisions.

The interoperability framework forms the backbone of the overall system architecture. It coordinates the communication between the simulation controllers, CARMONY services, and the simulation framework, ensuring a consistent exchange of information and enabling the integration of heterogeneous components developed by different project partners.

As described in deliverable D3.1, the CARMONY orchestrator is structured into three layers:

- Regulatory Layer
- Preventive Layer
- Reactive Layer

Each orchestrator layer imposes different requirements for the traffic simulation. The reactive layer requires simulation results within short timeframes and therefore relies on lower-fidelity models. In contrast, the preventive layer focuses on the assessment of traffic management strategies and can employ higher-fidelity simulations, where execution time is less critical. Depending on the evaluation objective, the regulatory layer may utilise either approach.

In addition, an environment simulation is required to emulate the traffic system in scenarios where real-world traffic data is not available. This enables the validation of orchestration functionalities in a fully simulated setting and supports the evaluation of CARMONY use cases under controlled conditions. Consequently, the simulation framework must support multiple simulation modes and fidelity levels to address the requirements of all orchestration layers and project use cases.

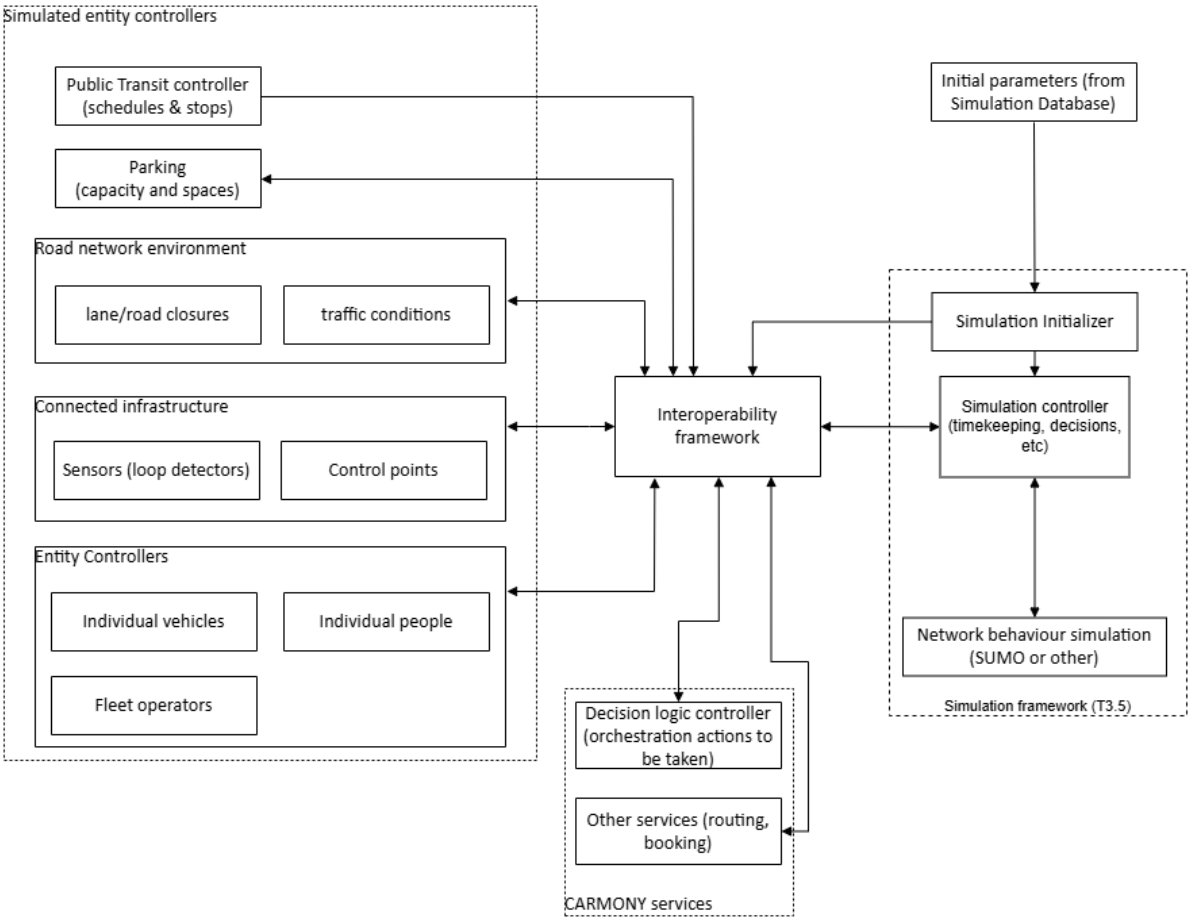

Figure 1: Simulation System Overview

3.1 Simulation Framework Requirements

The following requirements are relevant for the simulation framework. Their status is given as "implemented", "partially implemented", or "planned".

Table 1: Cross-cutting simulation requirements

ID	Description	Status
CCR_010	Scenario simulations are needed for system performance validation	implemented
CCR_022	The system shall support both real-time and non-real-time simulation modes for traffic scenarios.	planned
CCR_034	[ViF] Simulation scenarios shall be designed to explore the impact of varying connected and automated vehicle (CAV) parameters on traffic flow performance.	planned
CCR_035	The simulation framework shall support variation of the penetration rate of CAVs and CVs among conventional vehicles.	planned
CCR_037	A comprehensive analysis of the current traffic situation as a baseline for future improvements and evaluations.	implemented
CCR_040	[ViF] Simulations shall allow adjustable fidelity levels of vehicle simulation models to balance accuracy in replicated scenarios.	planned

ID	Description	Status
CCR_064	The Real-Time Traffic Orchestrator shall deliver simulations for real-time guidance to CAVs to support coordinated traffic flow.	planned
CCR_066	The scope and context of each simulation such as environment, input and output data, shall be defined prior to simulation execution.	implemented
CCR_067	Functional characteristics required for all tools shall be integrated into the simulation environment.	
CCR_089	The simulator should be able to model realistic traffic.	implemented
CCR_090	The simulation should be able to simulate connected vehicles.	planned
CCR_092	The penetration rate shall be set as % connected vehicles of the overall number of vehicles/overall traffic flow.	planned
CCR_093	The vehicles within the traffic network are represented as single actors without high fidelity multi body vehicle dynamics.	implemented
CCR_095	The simulation should be able to model different penetration values of different road users.	planned
CCR_096	The simulation should be able to model the connection between Orchestrator and connected (automated) vehicles and VRUs who use the Carmony-App.	planned
CCR_100	The simulation framework should be able to generate result files with travel time, traffic density in a human-readable format (preferable csv).	implemented
CCR_101	There should be a low fidelity simulation for use in the Orchestrator to assess situations not included in the database.	planned
CCR_136	Simulations shall be clearly defined in advance, including number of simulations to be conducted, number of traffic participants, required level of detail and accuracy, CVs penetration rate. All this information will be conveniently documented.	implemented
CCR_139	The number of traffic participants must be adjustable in the simulator.	planned
CCR_140	The simulator shall assume that it is given correct positions from CEVs.	implemented

Table 2: Use Case 1 simulation requirements

ID	Description	Status
UC1_011	The simulation shall reflect an urban environment within the city of Murcia.	planned
UC1_012	The simulation shall include a low emission zone and standard city zone.	planned
UC1_013	The simulation shall build on a road network showing relevant parts of the city of Murcia.	planned

UC1_015	The simulation does not include adverse weather conditions.	implemented
UC1_016	The simulation does not consider the quality of signal (signal loss) for connectivity.	implemented
UC1_017	The density of traffic should change according to daytime (commuting).	implemented
UC1_018	The city traffic simulation shall include passenger vehicles, heavy duty vehicles, cyclists, busses and pedestrians.	planned
UC1_019	The simulation shall not include a high fidelity model of automated driving functions.	planned
UC1_026	The simulation does not simulate the driver and the interaction with the app, but simplifies this by receiving and sending of messages.	planned
UC1_033	The simulator should be able to simulate Low Emission Zones (LEZs) restrictions.	planned
UC1_034	The simulator should be able to simulate multimodal traffic, e.g. using a private car – parking the private car – continuing the journey by public transport	planned
UC1_035	The simulator should be able to model vulnerable road users (VRUs) like pedestrians.	planned
UC1_036	The simulator should be able to model public transportation.	planned
UC1_039	In the simulation traffic sensors should be modelled.	planned
UC1_040	In the simulation, simulation of traffic, simulation of the orchestrator, and simulation of communication should be connected.	planned
UC1_042	The simulation should be able to model the connection between Orchestrator and connected (automated) vehicles and VRUs who use the Carmony-App.	planned
UC1_043	The simulation needs to take into account Murcia site special characteristics.	planned

Table 3: Use Case 2 simulation requirements

ID	Description	Status
UC2_013	The simulation shall reflect an urban environment within the city of Murcia.	planned
UC2_014	The simulation shall include an intersection where an incident blocks the road.	planned
UC2_015	The simulation shall build on a road network showing relevant parts of the city of Murcia.	planned
UC2_016	The simulation shall be able the block roads for the traffic within the road network.	planned
UC2_018	The simulation does not include adverse weather conditions.	implemented
UC2_019	The simulation does not consider the quality of signal (signal loss) for connectivity.	implemented

ID	Description	Status
UC2_020	The density of traffic should change according to daytime (commuting).	implemented
UC2_021	The city traffic simulation shall include passenger vehicles, heavy duty vehicles, busses, cyclists and pedestrians.	planned
UC2_022	The simulation shall not include a high fidelity model of automated driving functions.	planned
UC2_029	The simulation does not simulate the driver and the interaction with the app, but simplifies this by receiving and sending of messages.	planned
UC2_037	The simulator should be able to model vulnerable road users (VRUs) like pedestrians.	planned
UC2_038	The simulator should be able to simulate damaged connected (automated) vehicles.	planned
UC2_043	The simulation should be able to provide an alternative means of transport and simulate the fulfilment of the transport wish.	planned
UC2_044	The traffic simulation should contain a model of bus, which can have the state “broken down” and is connected.	planned
UC2_045	The simulated environment should describe the test region in Murcia respectively a part of it.	planned
UC2_046	The simulation should be able to model the connection between Orchestrator and connected (automated) vehicles and VRUs who use the Carmony-App.	planned
UC2_047	The simulation needs to take into account Murcia site special characteristics.	planned

Table 4: Use Case 4 simulation requirements

ID	Description	Status
UC4_039	The simulator should be able to model carpool and bus lanes.	planned
UC4_041	The simulator should provide data to compute KPIs like number of traffic jams, carpool lane utilization, number of accidents, fuel consumption, as results.	partially implemented
UC4_042	The simulated environment should describe the test region in Luxembourg respectively the necessary part of it.	implemented
UC4_043	The simulator should restrict the use of certain lanes to registered users.	planned
UC4_044	The simulator should model traffic counters and sensors.	implemented
UC4_048	The simulation does not simulate the driver and the interaction with the app, but simplifies this by receiving and sending of messages.	implemented

Table 5: Use Case 6 simulation requirements

ID	Description	Status
UC6_010	The simulator should be able to simulate variable speed limit signs (VSLs), where the speed limit is an input from the orchestrator.	implemented

4. Network Management Simulation Controllers

This chapter discusses work related to T3.4: Network management simulation controllers, presenting all progress on agent-based controllers for simulating the various functionalities of CARMONY within a network management simulation environment.

4.1 Overview

This section presents a high-level overview of the Network Management Simulation system. At a high-level the system is comprised of a collection of autonomous controllers and services, each of them emulating a specific system relevant to road traffic management, and an interoperability framework, that connects all of them together.

4.2 Network management controllers

All network management controllers are tasked with emulating a process or system relevant to traffic management, and each is set up in a similar manner: in most cases they contain a database that holds records of the emulated entities and their properties, and they provide a small set of relevant functionalities (such as retrieving or updating information) via an API. A summary of the network management controllers is provided in Table 6.

Table 6: Network management controllers list

Group	Controller name	Database content	Available queries
Public transit		Information about public transit schedules and stops	Get stops locations
			Get route schedules
Parking		Information about parking spaces and occupancy	Get parking space locations
			Get or set capacity and current occupancy for a parking space
Road network environment controllers	Road network	Graph representation of the road network	Get road network graph
			Get road segment characteristics (lanes, speed limits, etc)
	Traffic conditions	Information about current traffic conditions on road segments	Get current traffic conditions for a given road segment
	Closures	Info about current and planned closures and restrictions in the road network	Get all road segments that are under closures/restrictions currently
			Get all road segments that will be under closures/restrictions in near future
			Get information about a particular road segment
Connected infrastructure controllers	Sensors	Info about current traffic counts, simulating loop detectors or similar	Get or set latest traffic data for sensor
			Get past traffic data for sensor
	Control points	Locations and capabilities of communication and control points (signalling	Get control points by type within area

		systems, lane blockers, etc)	Get or set current state of control point
	Traffic lights	Locations and timings of traffic signals	Get current state of traffic light
			Get or set timing of traffic light
Entity controllers	Vehicles	Records and info of all relevant vehicles including occupancy, location, type, etc	Get or set current info about specific vehicle
			Get all vehicles by filter
	Users	Record of relevant individuals/VRUs	Get user characteristics and properties
			Get current information about user
	Fleet operators	Records of all vehicles in fleets	Get information about fleet

4.2.1 Public transit controller

The public transit controller holds information about public transit in a given urban environment. It can provide information about the public transit stops network, as well as route schedules.

4.2.2 Parking controller

The parking controller emulates the behaviour of parking space operators. It contains in its database information about the locations, capacity, and current occupancy of parking spaces, and provides endpoints for retrieving all of the above, as well as updating current occupancy for a given parking location.

4.2.3 Road network environment controllers

The road network environment group contains all controllers that are related to the road network itself.

4.2.3.1 Road network controller

The road network controller holds a representation of the road network in graph format. It provides endpoints for other systems to retrieve the network graph (e.g. for pathfinding or visualization operations), and also to retrieve additional information about a specific road segment.

4.2.3.2 Traffic conditions controller

The traffic conditions controller acts as an emulator for a traffic monitoring system. It holds a record of current traffic conditions throughout the transport network and provides an API endpoint for retrieving current conditions. It may be coupled with the sensors controller so that the two are in alignment during a given scenario.

4.2.3.3 Closures controller

The closures controller emulates the behaviour of a traffic disruptions monitoring system. It holds information about current and planned closures and disruptions in the road network and provides endpoints for retrieving this information in various formats: list of all currently restricted road segments, future planned restrictions for a given timestamp, or restrictions information about a particular road segment.

4.2.4 Connected infrastructure controllers

The connected infrastructure controllers group contains all controllers whose operations relate to emulating some part of networked road infrastructure.

4.2.4.1 Sensors controller

The sensors controller emulates the behaviour of traffic sensors (such as traffic cameras, loop detectors, etc) and holds information about the latest traffic conditions detected by each sensor. It provides endpoints for updating latest traffic data for a sensor (e.g. via simulation) and allows the retrieval of current or past traffic conditions detected by a sensor. Information uploaded to this controller may also be forwarded to the Traffic conditions controller to update the overview of the traffic conditions and keep the two in alignment.

4.2.4.2 Control points controller

The control points controller emulates the behaviour of control and communication infrastructure, for example variable road signs (VRS). It holds the positions and capabilities of each point and provides two endpoints: one for retrieving all control points by type in an area, and another for getting or setting the current state of a control point.

4.2.4.3 Traffic lights controller

The traffic lights controller emulates the behaviour of traffic lights at junctions. It holds the locations and timings of traffic signals and provides endpoints for reading the current state of a traffic signal and for getting or setting the timing of a traffic light.

4.2.5 Entity controllers

The entity controllers group contains all controllers that hold information about the active entities in a road network system.

4.2.5.1 Vehicles controller

The vehicles controller holds information about all relevant vehicles of interest in the network, including information such as current location, type of vehicle, occupancy, etc. The main purpose of this controller is to enhance the overall simulation capabilities of CARMONY by adding additional layers of information on top of the vehicle simulation, such as whether a vehicle is part of a fleet, without having to incorporate this information directly into the simulation engine. It provides an endpoint for updating or reading information about a specific vehicle, and another for retrieving all vehicles that fit search criteria (e.g. location, type).

4.2.5.2 Users controller

The users controller holds information about transport network users (e.g. commuters, travellers, VRUs etc). Its main purpose is to differentiate between vehicles and their occupants, for cases such as individuals needing to make use of multi-modal transport services, car-pooling, or similar, where the individual does not always map one-to-one with a vehicle. It provides endpoints for retrieving all information about a specific user.

4.2.5.3 Fleet operators controller

The fleet operators controller aims to emulate the behaviour or fleets. It holds a record of all fleet vehicles and any particular properties that the fleet may hold (such as special

permissions/restrictions). It provides an endpoint for retrieving information about the fleet or for specific vehicles in the fleet. The records held by this controller may be in relation to the vehicles controller: while the vehicles controller holds information about each vehicle, this fleet operators controller holds information about the vehicle fleet as a whole.

4.3 Interoperability framework

The Interoperability Framework acts as an interface between the various Network management controllers, simulation framework, and other CARMONY services. This allows each of the subsystems to operate more efficiently within the CARMONY architecture. More specifically, simulations are utilized for a variety of different purposes in CARMONY, including evaluation of mitigation measures in real-time, exploratory simulations to anticipate future scenarios, and evaluating long-term impacts of governance and regulatory decisions. Furthermore, there are multiple systems and processes relevant to the scenarios explored in CARMONY, including public transit, parking capacities, fleet operators and managers with human-in-the-loop capabilities, traffic detection systems and sensors, etc.

Given the range of scenarios covered in the six CARMONY Use Cases (UCs) as presented in deliverable D2.1, we anticipate that all of the above simulations and processes will need to be connected in different combinations depending on UC: For example we expect simulations to be deployed using real-time data from the real world in one case, while in others we expect simulations to be deployed offline in order to examine multiple potential solutions to the same incident and identify the best response, therefore requiring input from systems that are fully controlled, i.e. not real-time. In a similar manner, we will train our detection and optimization systems both on real-time data and synthetic data generated via simulations.

As such, rather than developing multiple versions of our tools in order to allow them to be able to interface with each other, we propose the introduction of the Interoperability Framework in the middle: Each system is designed with a single purpose and with a clearly defined set of inputs and outputs, and it is the Interoperability Framework's job to provide the data in the correct format. This means that, for example, the simulation framework only needs to interface with the Interoperability Framework. Afterwards a simulation can be fed the input data it requires, and its output data is forwarded to the system that requires it, without having to define the connections in the simulation framework explicitly. Furthermore, this approach allows for greater interchangeability and modularity, as one subsystem can be exchanged or upgraded without affecting the operation of others: since the only system that directly interfaces with all other systems is the Interoperability Framework, only it needs to adapt, while the rest of the systems will continue to operate as before.

In order to enable the architecture described above, the Interoperability Framework provides an API with a comprehensive list of endpoints available, covering all of the network management controllers and relevant systems in CARMONY. Furthermore, the Interoperability Framework handles data format conversions between the different subsystems, ensuring that each tool receives the input data in the format it expects, regardless of which service the data was sourced from. As such, essentially at its core the Interoperability Framework may be considered as an API with a pass-through pipeline to the various services, plus data converters attached to all endpoints to handle the necessary data transformations. An overview of the Interoperability Framework architecture is shown in Figure 2 below.

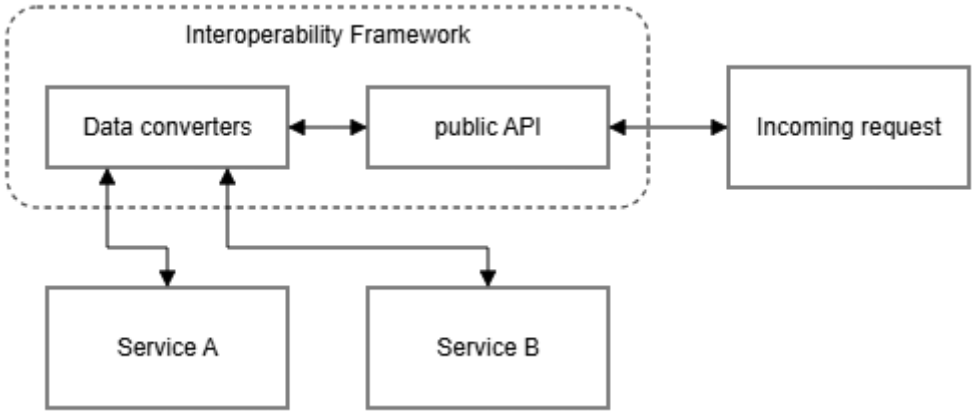

Figure 2: Interoperability Framework simplified data architecture

5. Simulation Framework

This section contains all work related to Task 3.5, focussing on the simulation framework for evaluating network management orchestration requirements.

5.1 Overview Components

Figure 3 provides an overview of the simulation framework. As discussed in Section 3, the simulation framework must address different simulation requirements. In particular, a distinction must be made between the traffic simulations used within the orchestrator layers and the environment simulation representing the traffic system itself. To accommodate these requirements, the simulation framework supports both an open-loop and a closed-loop approach.

The purpose of the traffic simulations within the orchestrator is to evaluate potential mitigation measures, such as opening a carpool lane or applying speed regulations, before they are deployed in real traffic or, in the case of a fully simulated scenario, before they are applied to the environment simulation. In these simulations, both the traffic conditions and the actions to be evaluated are known in advance. Once a simulation has been started, no additional actions are introduced. After completion of the simulation run, the results are analysed and key performance indicators (KPIs) are calculated. This corresponds to an open-loop simulation approach. It supports both low- and high-fidelity simulations, enabling the evaluation requirements of the reactive and preventive orchestrator layers, respectively.

In contrast, the environment simulation operates as a closed-loop system. During runtime, the current traffic state is continuously provided to the orchestrator, which may generate new control actions based on the observed conditions. These actions are then applied to the running simulation, creating a continuous feedback loop between the orchestrator and the simulated traffic environment.

For the closed-loop approach, two operating modes are considered. The first one is a step mode, primarily intended for development and debugging purposes. In this mode, the environment simulation is paused after each simulation step and waits for updated actions from the orchestrator before continuing. The second is an online mode, in which the simulation executes continuously and applies new actions whenever they become available. This mode is closer to real-world operation.

Both simulation approaches share a common three-layer architecture, consisting of an **API layer**, a **backend layer**, and a **simulation layer**. The responsibilities of these layers are briefly outlined below and described in detail in Sections 5.2–5.4.

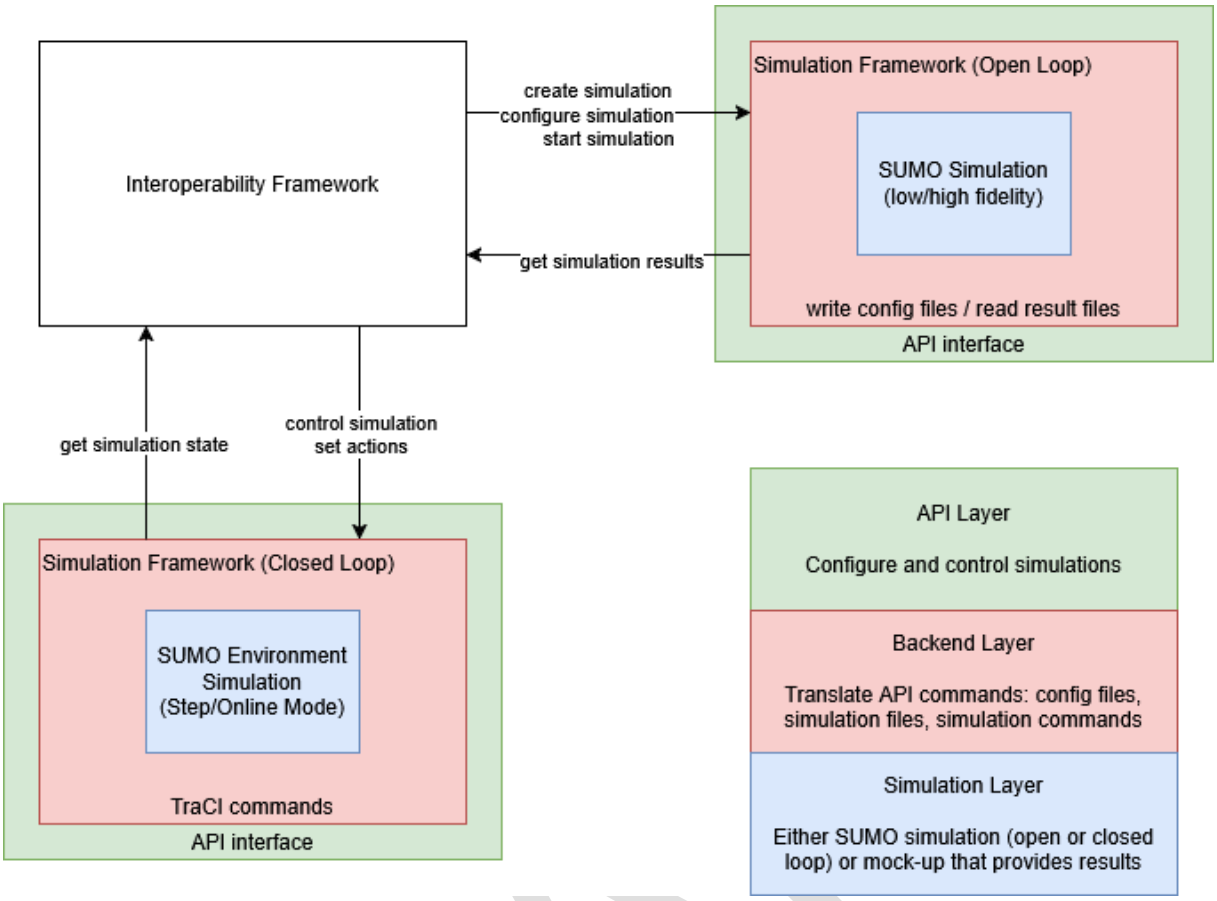

Figure 3: Overview Simulation Framework

5.2 API Layer

The API layer is the interface through which simulations are being set up, controlled, and their results are being read. The API interface for the open loop simulations allows creating and running simulation scenarios, and receiving the results of successfully finished simulation runs. The interface for the environment simulation adds additional commands to that to control the simulation during runtime and receive current traffic KPIs.

The API layer is implemented as a RESTful service using the FastAPI framework. Data is exchanged in JSON format, with schemas defining the available fields and their types.

5.3 Python/Backend Layer

Below the API layer, a python backend translates the API requests into files and commands. The used version is 3.14.5, with the required version being 3.12 or newer. The exact versions of all packages used are defined in the uv.lock file part of the software repository.

When creating a new simulation scenario, the simulation manager creates a directory containing all files needed for successfully running the SUMO simulation. The source of these files is the corresponding baseline of the simulation, meaning an already existing simulation (without orchestration) that the simulation is being compared to. Any orchestration actions (such as speed regulations) that are to be tested in simulation are being added in the SUMO configuration files. In some cases, such as speed recommendations being sent out to specific Connected Vehicles (CVs), this is not possible; instead, the TraCI (the traffic control interface) package is being used to implement these actions during simulation runtime.

When simulation results are requested, the SUMO output files are processed and the relevant data is aggregated. The results are resolved spatially and temporally, with a granularity specified at simulation creation.

In the case of the closed loop approach, current traffic data can be requested and is retrieved directly from the simulation environment via TraCI. The data is provided using the same spatial and temporal resolution as the simulation results. In the case of KPIs pertaining to specific vehicles, the results are resolved only over time, and are also being collected during runtime via TraCI.

5.4 Simulation Layer

The simulation layer consists of the actual SUMO simulation. It is being started through TraCI from the backend and runs in the background or visualised by the SUMO GUI. The used version of SUMO is 1.27.0. There are different types of simulation being used.

- The low- and high-fidelity simulations implement the open-loop approach introduced in Section 5.1. Results are being read after finishing. The differences between low and high-fidelity lie in the time step being used, the geographical extent of the map, and in using a pre-simulation phase to give more accurate results.
- The environment simulation runs closed loop, meaning that during runtime new commands can be given to react on the state of the simulation. It is meant to replicate real traffic, and thus runs continuously. There are two possible modes:
    - In Step Mode, the simulation progresses to a certain time step via a command and then remains there until the next one. This means that here traffic does not run in real-time, and that orchestration decisions and traffic situations can be analysed and evaluated without time constraints.
    - In Online Mode, the simulation progresses by itself by doing a one-second time step each second, running in real-time. The behaviour of the simulation is identical in both cases, but in online mode the performance of the orchestrator can be tested under realistic conditions.

5.5 Initialisation Process

Starting a simulation requires two steps. First, the simulation gets initialised with a `/simulations` API call. Then, a separate call (`/simulations/{sim_id}/start`) starts the simulation, as simulations might be created well in advance of running them. After finishing the simulation, results can be requested with `/simulations/{sim_id}/results`.

Each simulation belongs to a directory in the simulations folder, containing setup and result files. When initialising a simulation, the setup files (sumocfg file and additional file) are being copied from the baseline simulation and then adapted to implement the orchestration actions to be simulated. The sumocfg file contains paths to the network and demand files, which are dependent on the baseline. The additional file contains actions such as variable speed signs, as well as definitions where induction loops are positioned and along which edges KPIs are being recorded. The period with which these data are being aggregated and saved is also given by the additional file and can be configured at simulation initialisation.

6. Demo presentation

In order to showcase the functionalities of the CARMONY simulation system discussed in the previous chapters, we have designed a demonstration that runs through some of its processes. This demonstration acts as a vertical slice, containing a subset of the simulation system's functionalities that we consider to be indicative of its capabilities. We have opted to demonstrate the simulation system by running a scenario relevant to one of the use cases of the project, that of the speed reduction measures in the A3 highway in Luxembourg (UC 6: Speed Reduction for Daily Traffic Jam Avoidance, as outlined in CARMONY deliverable D2.1).

6.1 Demonstration setup

For the purposes of this demonstrator, we implement a few auxiliary processes to help with the execution of the overall simulation process. We furthermore include some assumptions in the whole simulation process. Both of these steps have been adopted only for this initial version of the simulation system and specifically for the demonstrator presented in this deliverable, and they are not part of the actual CARMONY simulation system. We describe these in more detail here, to help focus on the actual functionalities of the simulation system.

6.1.1 Assumptions in the demonstrator

The simulation process presented here showcases the capabilities of the system in a known scenario of the CARMONY project, that of speed regulation measures on the A3 highway in Luxembourg. We assume that the following have taken place, prior to the simulation being executed: First, that a traffic degradation has been predicted, and thus the need for a mitigation measure has been identified (e.g. from tools developed in Task 3.2). Second, that a mitigation measure has been calculated (e.g. as produced by tools from Task 3.3), which recommends speed reduction regulations for a short duration. Third, we assume that both the initial traffic conditions and the mitigation measures are available in a machine-readable format (from the Simulation Situations Database and from the Decision Logic blocks, respectively) and can be fed to the simulation system.

6.1.2 Auxiliary processes

For the purposes of this demonstrator, we have opted to hard code some of the elements within the demonstrator, by implementing some additional functionality. First, we introduce a single demonstrator script that will control and orchestrate the whole demonstration process and will directly make any execution and query calls to the required endpoints. This will include providing the initial traffic situation parameters to the simulation (emulating the situations database), performing a call to the Interoperability Framework endpoint to retrieve any relevant mitigation measures, and making the necessary calls to the simulation controller to load, setup, and execute the simulation, and retrieve the simulation results at the end.

In addition to the above controller script, we also highlight here that the mitigation measures in the demonstration will not be produced in real-time by the Decision Logic block. Instead, we have pre-calculated a set of speed reduction measures. This set of mitigation measures is loaded with the Interoperability Framework system and is available through a special endpoint that will always return only this set of mitigation measures. We are therefore performing the actual call from the simulation system to the Interoperability Framework to retrieve any mitigation measures, but then skip the retrieval of the mitigation measures from the Interoperability Framework to the Decision Logic block and emulate this functionality instead by returning the pre-calculated set of mitigation measures.

We reiterate here that all of the auxiliary functionality presented here is only implemented for the purposes of this demonstrator and is not part of the actual CARMONY simulation system. In the final version the various systems are expected to communicate with one another and trigger the various functionalities as needed, e.g. the traffic monitoring tools continuously evaluate real-time data to anticipate traffic degradations, when such are detected appropriate mitigation measures are calculated and starting traffic conditions are stored, a request is made to fire up a high fidelity traffic simulation to evaluate the potential impact, etc, all without the need of an auxiliary controller script.

6.2 Demonstration description

The demonstration process involves the execution of a simulation scenario using the CARMONY simulation system. This was executed on one computer with the whole process being screen recorded. The resulting video accompanies this report and is also available to view at https://youtu.be/_IDjI2UTR1A. The rest of this section describes what happens during the video and provides additional information where needed.

The demonstration includes three main processes: The Simulation Framework and Interoperability Framework that we have presented in this document, and the demonstration controller process that we have set up specifically for the purposes of directing this demonstration. Both the Simulation Framework process and the Interoperability Framework process are running as independent services, each one exposing its functionality via an HTTP API. Both processes are running locally, with their APIs available at '`http://127.0.0.1:8000`' and '`http://127.0.0.1:8001`', respectively. The demo controller runs as a separate process, making calls to the respective API endpoints as needed. The three main processes are shown in Figure 4.

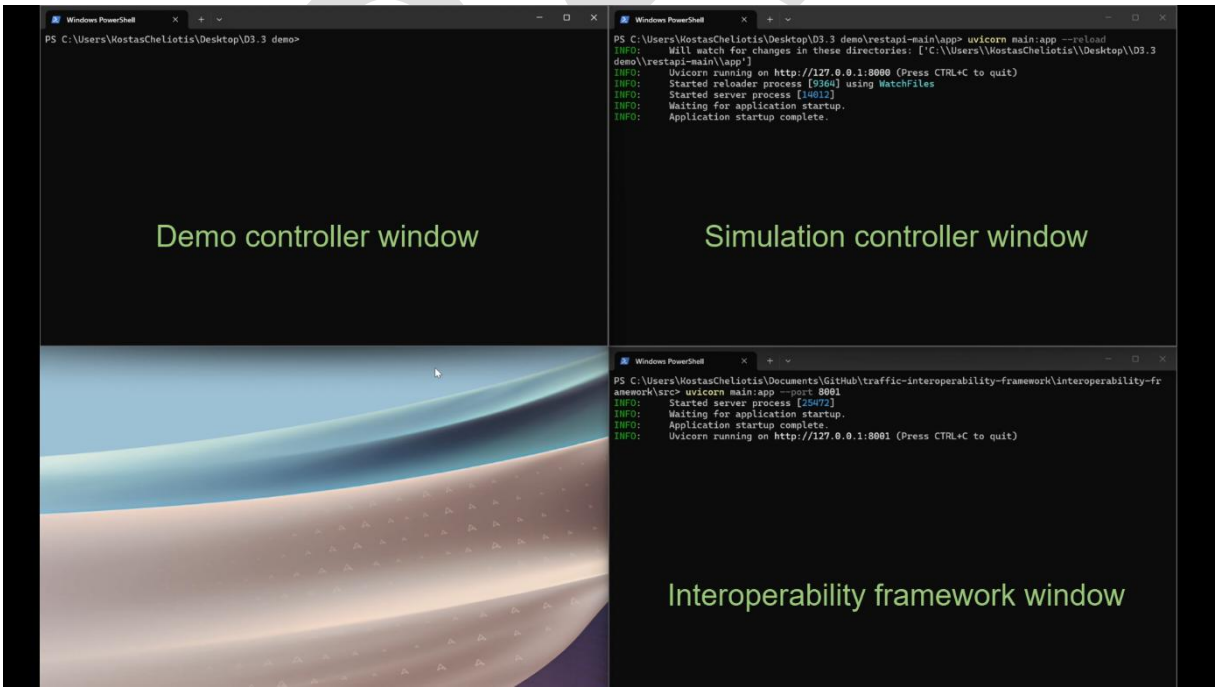

Figure 4: The demo controller process, Simulation controller, and Interoperability framework running as three separate processes

First, the demo controller makes a request to the Interoperability Framework API to retrieve the calculated mitigation measures to apply to the scenario, via the endpoint `'http://127.0.0.1:8001/mitigation_actions/latest'`. This returns a speed regulation in JSON format, shown in Figure 5. This mitigation measure involves a suggestion to set a temporary speed limit for the length of the highway between control points with IDs "9438" and "11397", for a 15-minute duration between 08:00 and 08:15 am.


```
{
  "speed_regulation": 60,
  "start_pk": "9438",
  "end_pk": "11397",
  "start_time": "08:00",
  "end_time": "08:15"
}
```


Figure 5: Mitigation measure sample response for a speed regulation

Once the mitigation measure is returned by the Interoperability Framework, the demo controller builds the configuration file that will contain all of the scenario parameters for this particular simulation. These include a simulation id, a baseline scenario to build off of, the simulated time and duration of the simulation, the extent of the highway to be simulated, as well as the speed regulations to apply. A sample of the simulation configuration file is shown in Figure 6, that also includes the speed regulations that were just received by the Interoperability framework.


```
{
  "id": "demo_sim",
  "seed": 42,
  "baseline": "workday",
  "regulations": [
    {
      "speed_regulation": 60,
      "start_pk": "9438",
      "end_pk": "11397",
      "start_time": "08:00",
      "end_time": "08:15"
    }
  ],
  "start_pk": 110,
  "end_pk": 13332,
  "start_time": "07:00",
  "duration": 120,
  "time_resolution": 15,
  "kpi_list": []
}
```


Figure 6: Sample simulation configuration file

When the simulation configuration file is ready, it is passed to the Simulation controller API to create and set up the simulation, by making a POST request to the endpoint '`http://127.0.0.1:8000/simulations`' and passing the simulation configuration in the POST body. The Simulation controller API responds with a code 200 signifying that the simulation was created successfully and also returns the id of the simulation that was just created. At this point everything is set up and the simulation can be executed.

In order to run the simulation, a separate call needs to be made to the Simulation controller API using the endpoint '`http://127.0.0.1:8000/simulations/{sim_id}/start`', where the parameter `{sim_id}` contains the id of the simulation to be executed. Once the API call is made to this endpoint using the correct simulation id ('`http://127.0.0.1:8000/simulations/demo_sim/start`' in this case), the Simulation controller begins the simulation process. This involves initializing the simulation engine (in CARMONY we are using the SUMO engine) and passing the simulation configuration file to the simulation engine. A frame from the demonstration video is shown in Figure 7 that includes a view of the simulation window showing the synthetic vehicles in the simulated environment. One thing to note here is that we have only included the GUI view of the simulation engine for the demonstration video; under normal operation the simulations run without a visual representation, for optimized performance.

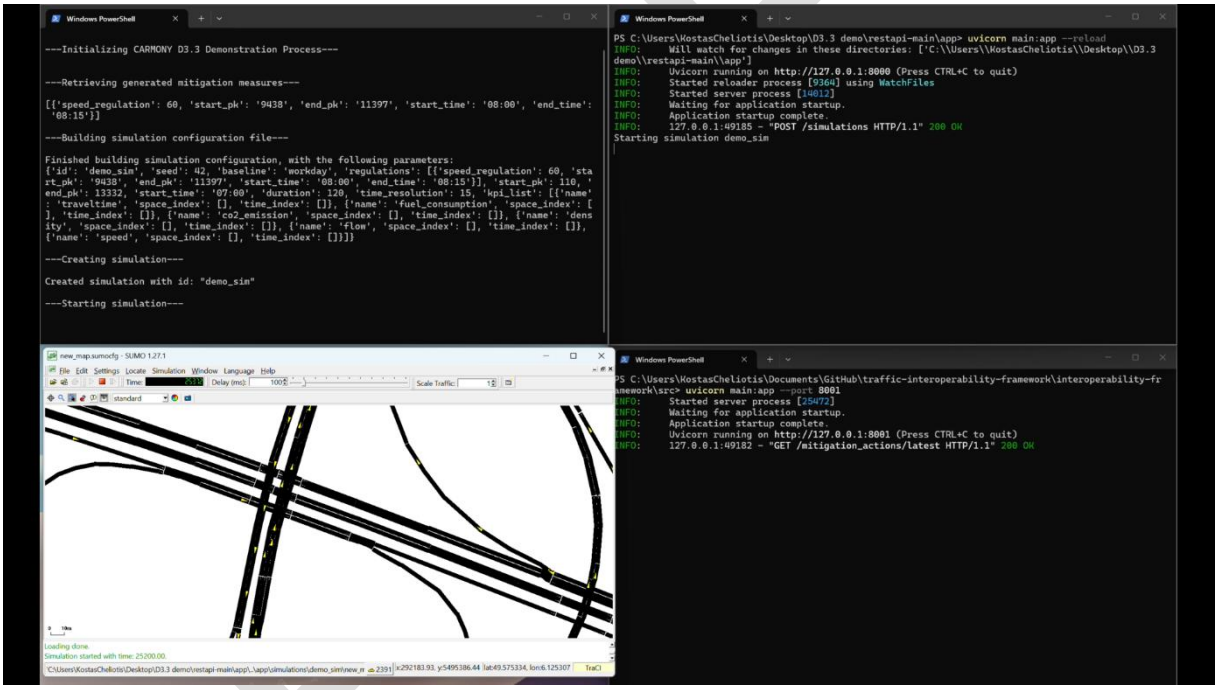

Figure 7: Snapshot of the demonstration video with the simulation engine in view (bottom left)

Once the simulation is finished, the demo controller makes one more call to the Simulation controller API to retrieve the simulation results. These are available via the API endpoint '`http://127.0.0.1:8000/simulations/{sim_id}/results`' only after the simulation has concluded. The simulation results are returned in JSON format, and include lists of metrics (such as average speed, flow, and density) captured during specified timesteps in the simulation. A snippet of the simulated flow results is included in Annex B for reference, providing a value per stretch of highway between control points, per 15 minutes of simulated time.

These results can then be used in further analysis or by stakeholders to evaluate the effectiveness of the mitigation measure originally provided by the Decision Logic. With this final step, the simulation process concludes, and the demonstration is completed.

7. Conclusion

This deliverable has presented the progress so far regarding the work of Tasks 3.4 and 3.5 of the CARMONY project, that focus on designing and developing the various systems that will support the simulation needs of the project. The Simulation system presented here has taken into account inputs, needs, and requirements as identified in previous work (deliverable D3.1) and has developed an initial working version of the simulation system that will be able to handle the varying simulation needs of the CARMONY system. In order to demonstrate these achievements, it has showcased the simulation system in action applied to a sample scenario from one of the CARMONY Use Cases, which can be seen in the accompanying video material that complements this report.

More specifically, in this report we have first highlighted how the work undertaken in this deliverable addresses the needs of other WP3 Tasks and the overall WP3 objectives. These are identified in how the simulation framework will provide a testbed for the calibration and evaluation of Task 3.2 and 3.3 systems, its support for interoperable systems and services, and its capacity to examine complex orchestration scenarios.

Following that we have outlined the four major components that comprise the CARMONY simulation system, namely the Network Management Simulation Controllers, the Simulation Framework, the Interoperability Framework, and the CARMONY Services. Out of these four components, three are the focus of this deliverable and have been discussed in more detail in Sections 4 and 5, providing an overview of their architecture and aims.

Having presented each of the major components in more detail, we then outline how they all come together to form the overall CARMONY Simulation System. To further demonstrate the feasibility and capabilities of it, we provide an illustrative example of the Simulation System, in the form of a video showcasing the system in action, that highlights its main functionalities.

The next steps will focus on four major goals: First, the initial working version of the CARMONY Simulation System will be evaluated in the context of the overall CARMONY Orchestrator and will inform and help shape the final version of its architecture that will be presented in future deliverable D3.4. Second, we will continue the development of the various subsystems needed to reach feature completion, as some elements are still under development at this stage of the initial version. Third, the CARMONY Simulation System will be deployed for the needs of the pilots, covering the simulation needs of the Use Cases and providing feedback on the Simulation System design. And fourth, based on the feedback gathered from the pilots, the various systems will be refined into their final version that will be presented in future deliverable D3.6.

8. Abbreviations

Abbreviations of terms used within the deliverable, in alphabetical order.

Term	Definition
API	Application Programming Interface
CARMONY	Coordinated Action for Responsive Mixed Orchestration and Network Yield
CV	Connected Vehicle
DEC	Websites, Patent Filings, Videos etc.
DEM	Demonstrator, Pilot, Prototype
GUI	Graphical User Interface
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
PU	Public
R	Document, Report
REST	Representational State Transfer
SEN	Sensitive
SUMO	Simulation of Urban MObility (traffic simulation engine)
TraCI	TRAffic Control Interface
UC	Use Case
VRS	Variable Road Sign
VRU	Vulnerable Road User
WP	Work Package

9. References

[1] “CARMONY D3.1: Specifications and conceptual architecture - Initial Version, V1.0 Final”, 2026-04-28

[2] “CARMONY Grant Agreement Number AMD-101202858-6”, 2026-05-22

[3] “CARMONY D2.1: CARMONY Use Cases refinement and specifications definitions, V0.6 Final”, 2025-10-27

A. Link to demonstration video

The video file that accompanies this report is also available at https://youtu.be/_IDjI2UTR1A

B. Sample simulation results


```
"flow": {
  "results": [
    [
      1258.8216666666667,
      3545.5919999999996,
      3068.2444444444445,
      2057.4900000000002,
      1279.8799999999999,
      2243.7250000000004,
      3548.7660000000005,
      4139.82,
      2437.8375,
      2465.21,
      1966.745,
      1988.378,
      3943.406,
      4095.0066666666667,
      3433.47375,
      4548.33,
      4037.4375,
      3394.22125
    ],
    [...],
    [...],
    [...],
    [...],
    [...],
    [...],
    [...]
  ],
  "index_space": [
    "110->1482",
    "1482->2720",
    "2720->8246",
    "8246->9438",
    "9438->10437",
    "10437->11397",
    "11397->12526",
    "12526->13332",
    "13332->12542",
    "12542->12526",
    "12526->11397",
    "11397->10437",
    "10437->9438",
    "9438->8246",
    "8246->4200",
    "4200->2720",
    "2720->1482",
```

```
    "1482->110"
  ],
  "index_time": [
    "07:15",
    "07:30",
    "07:45",
    "08:00",
    "08:15",
    "08:30",
    "08:45",
    "09:00"
  ],
  "unit": "v/h"
}
```


The simulation results are available as JSON data. The same data can be seen in Table 7.

Table 7: Sample simulated vehicle flow results

flow (v/h)	07:15	07:30	07:45	08:00	08:15	08:30	08:45	09:00
110 → 1482	1044	580	22	22	22	18	13	33
1482 → 2720	3379	1498	75	339	380	51	34	403
2720 → 8246	3179	419	741	811	261	182	804	499
8246 → 9438	1988	410	1465	1328	577	1552	621	1679
9438 → 10437	1203	776	1831	1548	1804	2159	1392	2109
10437 → 11397	2273	2172	3362	2907	3203	3377	3049	3168
11397 → 12526	3608	3487	4313	4190	4304	4282	4328	4253
12526 → 13332	4160	4099	4616	4541	4621	4495	4705	4481
13332 → 12542	2178	153	112	76	53	47	39	41
12542 → 12526	2078	289	135	24	19	28	29	37
12526 → 11397	1607	365	161	57	52	50	28	47
11397 → 10437	1302	622	236	80	65	63	40	64
10437 → 9438	3456	1545	440	109	48	72	44	146
9438 → 8246	4017	1264	433	105	67	67	50	160
8246	3357	1865	383	671	234	542	373	706

→ 4200								
4200 → 2720	4172	2499	510	2213	854	1734	760	2094
2720 → 1482	2881	1792	691	1535	1001	1101	1011	1418
1482 → 110	1658	1468	1565	1629	1514	1594	1589	1591