D3.4: URBANE Innovation Transferability Platform

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No. 101069782

©URBANE 2022

Project & Document Information

Grant Agreement No	101069782	**Acronym**	URBANE
Project Full Title	UPSCALING INNOVATIVE GREEN URBAN LOGISTICS SOLUTIONS THROUGH MULTI-ACTOR COLLABORATION AND PI-INSPIRED LAST MILE DELIVERIES		
Call	HORIZON-CL5-2021-D6-01		
Topic	HORIZON-CL5-2021-D6-01-08	**Type of action**	IA
Coordinator	INLECOM INNOVATION		
Start Date	01/09/2022	**Duration**	42 Months
Deliverable	D3.4	**Work Package**	WP3
Document Type	OTHER	**Dissemination Level**	PU
Lead beneficiary	INLE		
Responsible author	Ioanna Fergadiotou		
Contractual due date	[31/08/2025]	**Actual submission date**	[29/08/2025]

Disclaimer and Acknowledgements

Funded by the European Union

This project has received funding from the European Union's Horizon Europe research and innovation programme under Grant Agreement No 101069782

Disclamer

Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or CINEA. Neither the European Union nor the granting authority can be held responsible for them.

While the information contained in the document is believed to be accurate, the authors or any other participant in the URBANE consortium make no warranty of any kind regarding this material including, but not limited to the implied warranties of merchantability and fitness for a particular purpose.

Neither the URBANE Consortium nor any of its members, their officers, employees, or agents shall be responsible or liable in negligence or otherwise howsoever in respect of any inaccuracy or omission herein.

Without derogating from the generality of the foregoing neither the URBANE Consortium nor any of its members, their officers, employees, or agents shall be liable for any direct or indirect or consequential loss or damage caused by or arising from any information advice or inaccuracy or omission herein.

Copyright message

©URBANE Consortium. This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation, or both. Reproduction is authorised provided the source is acknowledged.

Deliverable Contributors	
Contributor Name	**Organisation (Acronym)**
John Limaxis	Inlecom Innovation (INLE)
Dimitris Rizopoulos	Inlecom Innovation (INLE)
Maria Kampa	Inlecom Innovation (INLE)
Thanos Karydis	Inlecom Innovation (INLE)
Harris Niavis	Inlecom Innovation (INLE)
Fotis Michalopoulos	Inlecom Innovation (INLE)
Aristea Zafeiropoulou	Konnecta Systems (KNT)
Anastasios Kakouris	Konnecta Systems (KNT)
George Misiakoulis	Konnecta Systems (KNT)
Konstantinos Kaldis	Konnecta Systems (KNT)
Maria Koutsoukou	Konnecta Systems (KNT)
Kostas Zavitsas	VLTN

Version History				
Version	**Date**	%	**Changes**	**Author**
0.1	05/07/2023	5%	Wrote the ToC	Maria Kampa (INLE)
0.2	14/01/2024	30%	Added diagrams and content in chapters 2, 3	Maria Kampa (INLE), Aristea Zafeiropoulou (KNT)
0.4	02/04/2024	50%	Extended internal report by adding a section on Application development.	Aristea Zafeiropoulou (KNT), Maria Kampa (INLE), Kostas Zavitsas (VLTN) Ioanna Fergadiotou (INLE),
0.6	21/3/2025	70%	New chapters 5 and 6, updates in all chapters	Ioanna Fergadiotou (INLE), Aristea Zafeiropoulou (KNT)
0.8	03/07/2025	80%	Final Sections Updates	John Limaxis (INLE)
0.9	20/08/2025	90%	Internal Peer Review	Pierre Van Wolleghem (NORCE)
1.0	29/08/2025	100%	Small changes	Ioanna Fergadiotou (INLE)

Quality Control (includes peer & quality reviewing)			
Date	**Version**	**Name (Organisation)**	**Role & Scope**
[01/08/2023]	0.1	Efstathios Zavvos (VLTN)	QM ToC Approval
[10/04/2024]	0.4	Yasanur Kayikci (VLTN), Maria Kampa(INLE)	50% Approval
[04/08/2025]	0.8	Pierre Van Wolleghem (NORCE)	Peer Review
[22/08/2025]	0.9	Yasanur Kayikci (VLTN)	QM Approval
[22/08/2025]	0.9	John Limaxis (INLE)	PM review
[29/08/2025]	1.0	Ioanna Fergadiotou (INLE)	Project Coordinator approval

Executive summary

This deliverable report describes the work carried out and results achieved under URBANE *Task 3.6 Digital Twins Infrastructure & Open Model Library*.

In particular, the report presents three key elements: a) the architectural aspects of the URBANE Digital Twin Platform, (b) the integration of the final versions of the URBANE models and other WP3 services and (c) the development of the CitIQore application. In addition, beyond the technical specifications and technologies used to develop the URBANE Digital Twin platform, its business applicability and use cases are showcased. The overall goal is to provide the consortium and Living Lab partners with a set of digital tools and assets that enables them to simulate innovative last-mile logistics activities and optimise their operations.

The report begins by setting out the functional and non-functional requirements of URBANE's Digital Twin Platform for last-mile logistics operations, as well as the specifications of the different models that were important for integrating them into URBANE Digital Twin platform. Furthermore, it presents the high-level architecture of all URBANE services and solutions that together form the URBANE Innovation Transferability Platform. A key focus was to use an open-source technology stack and a compatible architecture that allow smooth communication and integration between the Digital Twin and other URBANE digital assets (e.g. smart contracts or the Impact Assessment Radar).

Furthermore, the technical architecture of the URBANE Platform and its underlying Big Data infrastructure is then presented, while emphasis is placed on the Model Library of the project. More specifically, the report explains the model integration and containerisation process as well as the execution environment put in place to execute the WP3 models as standalone models or as a sequence of models for enhanced insights.

Finally, the report concludes with the design and development of the CitlQuore app — a user-friendly, domain-specific application that serves as the frontend of the URBANE Digital Twin Platform, enabling users to simulate different what-if scenarios related to logistics and last-mile interventions by executing sequences of models developed in WP3. The report also summarises the main conclusions of Task 3.6, highlighting the transferability aspects of the developed solutions.

Contents

List of figures

List of tables

Glossary of Terms and Acronyms

Acronym / Term	Description
ABM	Agent-Based Modelling
ADV	Autonomous Delivery Vehicles
AI	Artificial Intelligence
BPMN	Business Processing Modelling Notation
CBA	Cost Benefit Analysis
DT	Digital Twin
EPC	Electronic Product Code
ICT	Information Communication Technology
IAR	Impact Assessment Radar
KPI	Key Performance Indicator

LL	Living Lab
LDT	Local Digital Twin
LSP	Logistics Service Provider
LLx	Living Lab number
MEE	Model Execution Engigne
ML	Machine Learning
RFID	Radio Frequency Identification
UI	User Interface
OR	Operations Research
OD	Origin Destination
T&L	Transport & Logistics
PI	Physical Internet
VRP	Vehicle Routing Problem
WPx	Work Package number

1 Introduction

URBANE aims to accelerate the transition toward resilient, sustainable, and efficient last-mile logistics by equipping cities and logistics stakeholders with advanced digital tools and models for scaling innovative last mile deliveries solutions. To support this goal, Work Package 3 focuses on developing the project's core ICT framework and a set of innovation transferability tools that enable data-driven planning, scenario testing, and impact assessment. These tools include (a) the Digital Twin Platform, incorporating the WP3 models, (b) the CitIQore application, (c) the Impact Assessment Radar, and (d) the Blockchain infrastructure. Together, they form the URBANE Innovation & Transferability Platform, as illustrated in Figure 1-1. At the heart of this entire offering lies the work delivered by T3.6 - namely, the Digital Twin Platform and CitIQore – which are the primary focus of this report.

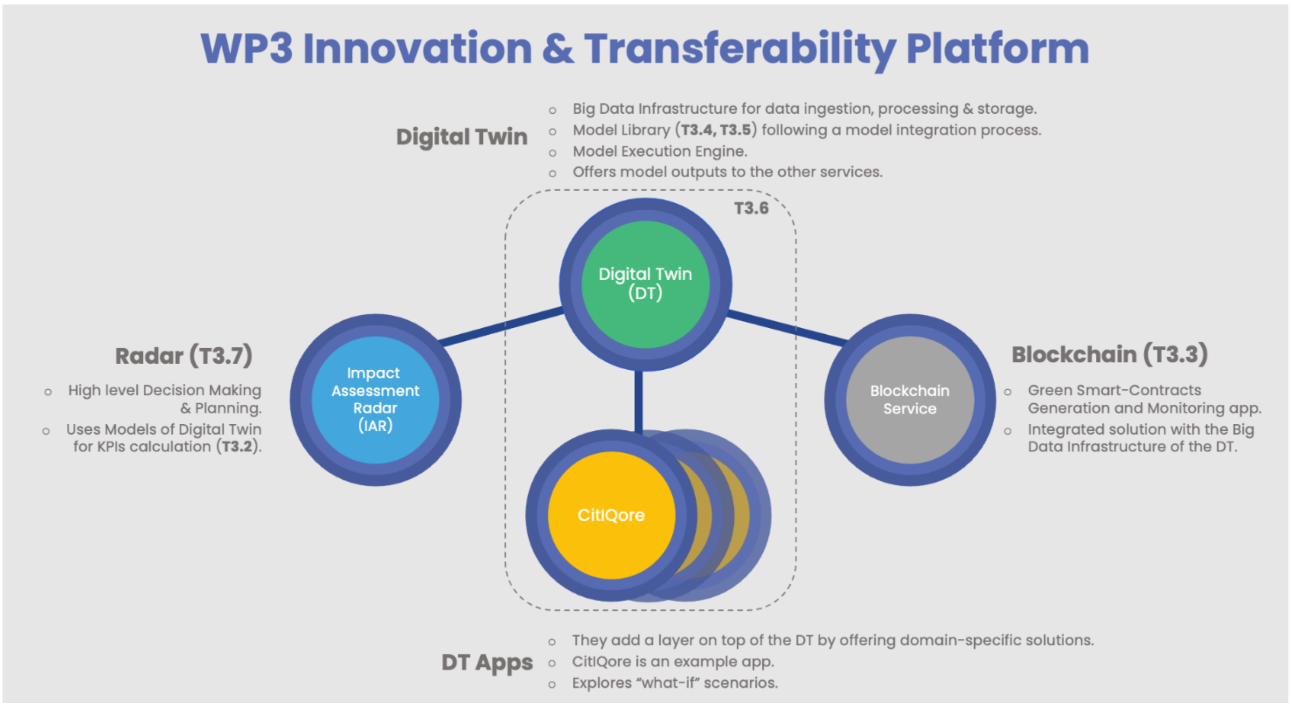

WP3 Innovation & Transferability Platform
Digital Twin
Big Data Infrastructure for data ingestion, processing & storage.
Model Library (T3.4, T3.5) following a model integration process.
Model Execution Engine.
Offers model outputs to the other services.
T3.6
Digital Twin (DT)
Radar (T3.7)
High level Decision Making & Planning.
Uses Models of Digital Twin for KPIs calculation (T3.2).
Impact Assessment Radar (IAR)
Blockchain Service
Blockchain (T3.3)
Green Smart-Contracts Generation and Monitoring app.
Integrated solution with the Big Data Infrastructure of the DT.
CitIQore
DT Apps
They add a layer on top of the DT by offering domain-specific solutions.
CitIQore is an example app.
Explores "what-if" scenarios.

FIGURE 1-1. WP3 INNOVATION & TRANSFERABILITY PLATFORM

Digital Twins (DTs) are virtual replicas of physical logistics operations that combine real-time data, advanced modelling, and clear visualisation to simulate and assess different scenarios. They leverage modern technologies such as containerisation, cloud computing, and secure data integration to ensure flexibility, scalability, and ease of deployment in various contexts. The URBANE Digital Twin Platform aspires to improve decision-making by enabling real-time data integration and simulations of transport and logistics actors and stakeholders. To achieve this, it comprises three main components:

- Simulation & Modelling: models that are integrated into the DT, thus empowering the platform with planning, optimisation and simulation capabilities.
- Real-time Data Processing & Analytics: The infrastructure supports the ingestion of large volumes of real-time and historical data. Event tracking enables dynamic, real-time decision making.

- Visualisation & User Interface: Through the CitIQore application, users can explore "what-if" scenarios to support decision making. The CitIQore map interface offers a digital representation of testing scenarios, while dashboards provide planning and reporting capabilities.

The following sections will outline the functional and non-functional requirements guiding the development of the Digital Twin Platform, describe its technical and conceptual architecture, and explain how Living Lab (LL) needs and data are incorporated. The report also presents the main implementation steps, including the development of the real-time data pipeline and the integration of the WP3 models that enable advanced simulations.

In addition, the report showcases the CitIQore application that has been technically designed and developed as an intuitive interface for configuring scenarios, running simulations, and visualising model outputs on top of the Digital Twin Platform. Finally, the report highlights how these technical solutions address the URBANE LL business needs through demonstration of the different technical solutions in real world example as well as their contribution to the broader digitalisation of transport and logistics systems by enabling detailed measurement, analysis, and evaluation of different measures and their impact on CO_2 reduction.

1.1 URBANE Outputs Mapping to GA Commitments

TABLE 1. DELIVERABLE ADHERENCE TO GRANT AGREEMENT DELIVERABLE AND WORK DESCRIPTION.

URBANE GA Item	URBANE GA Item Description	Document Chapter(s)	Justification
DELIVERABLE			
D3.4 URBANE Innovation Transferability Platform	*Dynamic data-driven DT System managing the coordination of models and data, interfacing to digital platforms, APIs and sensors; incudes system from T3.1 architecture considering scalability, elasticity, availability and connectivity requirements.*	*All of the D3.4 Chapters.*	*The report describes the main implementation of the Digital Twinning platform that has been carried out in Task 3.6. Chapter 2 presents the requirements for the development of the Platform, while Chapter 3 presents the conceptual architecture based on the work undertaken in Task 3.1. Chapter 4 presents the technical architecture, the infrastructure, and the Model Library. Chapter 5 presents the CitIQore application.*
TASK			
T.3.6 Digital Twins Infrastructure & Open Model Library	*Provide an integrated Modelling, Simulation and Optimisation Environment, integrating the models from Tasks 3.4 and T3.5.*	*Chapter 4*	*The URBANE Model Library is described in section 4.6, while the modelling and simulation environment in section 4.2.*

ST.3.6.1 Open DT modelling and simulation environment setup	*Setup and configurate a modelling and simulation environment, built on open standards (e.g., Functional Mock-up Interface standard) and open simulation frameworks (e.g., OSP , Red Flux), and key components of the LEAD DT platform to: (a) enable a unified representation of LLs settings, including logistics and transport assets and systems; (b) enable the re-use of models and DTs across LLs; (c) connect external models and control systems, enabling large scale co-simulation for virtual systems integration; (d) wrap all models as building blocks, for example as microservices or Functional Mock-up Units (FMUs), (e) virtualise last mile operations throughout a period of time, and (f) simulate PI-inspired urban logistics operations (as per outcomes of T1.4 with the contribution of KLU).*	*Section 4.2, Chapter 5*	*Section 4.2 discusses the open DT modelling and simulation environment setup, which is responsible for the execution of digital models of different types (i.e., simulations, optimization, AI).* *Chapter 5 presents the CitIQore app, allowing models to be reusable and transferable in any city context.*
ST.3.6.2 AI Tools and Big Data Analytics Infrastructure	*Provide data processing (e.g., Apache Flink and Spark) and storage (e.g., Apache Hadoop and Kafka, Cassandra, MongoDB), to support semi-automatic script-based data processing pipelines over distributed data sources, as well as AI analytics components, enabling the discovery and synthesis of data-driven innovative services. Support processing and information extraction from geospatial and time series data. Integrate AI components for UI-driven model management with training capabilities (supporting transfer learning), data analytics visualisation, and model/condition specific workflows creation for exploratory data analysis of new datasets. Support automated data capture, processing, cleansing, transformation and event-based data flows management for better visibility, performance, and enhanced automation of operations. Implement components for time series processing and analysis, including forecasting, as well as complex event processing by combining data from multiple sensors. Manage event-based data streaming components from the Collaboration Governance Ledger developed in WP3.*	*Section 4.5*	*Section 4.5 presents the steps followed to develop the necessary infrastructure that ingests, stores, maintains and serves data through the Big Data infrastructure.*

ST.3.6.3 Virtual Model Execution Engine - Simulations Orchestrator	*The engine will facilitate the representation of operational workflows, while an URBANE rule-based model will provide reasoning and evaluation capability by using rules extracted from historical data (e.g., operational performance data) and logistics operators' experience. The provision monitoring, analysis, simulation, optimisation, and prognosis for URBANE services will be supported.*	*Section 4.3*	*Section 4.3 presents the main developments regarding the Platform's Model Execution Engine, including the technical framework employed in that process.*
ST.3.6.4 IoT infrastructure and operational systems Connectors	*Develop the necessary connectors and APIs to setup the links between the LL's IoT infrastructure and external data resources. Develop connectors for operational systems based on the LLs use cases requirements. APIs developed and deployed in the context of the LLs will become available via an API Registry. APIs and metadata descriptions will be semantically enriched, using the modelling framework (e.g., ontology) and open library of reusable models developed in Task 4.2 to enable and facilitate semantic search and interoperability across the LLs.*	*Section 4.4*	*Section 4.4 focuses on the connectors developed to enable the ingestion of data from LL data sources. These serve the Blockchain use cases of Task 3.3 and the e-bikes intervention in CitIQore (section 5.3).*
ST.3.6.5 Containerization and integration of models developed in the LLs – Open library of reusable models	*Containerization and integration of models developed in the LLs - Open library of reusable models. Design the model execution workflows for the LLs, defining the sequence of models' execution, their inputs and outputs, and configuration parameters. Develop and deploy specific APIs (wrappers) for the models developed in the LLs, enhanced and semantically enriched via metadata. Integration with the platform's Execution Engine supported by INLE and KNT. Develop an open library and populate with ready-to use models stemming from Lighthouse LLs and EU-wide pilots that can be employed in DTs.*	*Section 4.2.2, section 4.6*	*Section 4.6 presents the Model Library, as well the model integration process that we follow in URBANE, while section 4.2.2 presents the containerisation process.*

1.2 Deliverable Overview and Report Structure

The deliverable is structured as follows:

- Chapter 2 outlines the requirements during the first months of the project, as well as those emerged following feedback from LL partners.
- Chapter 3 focuses on the conceptual architecture of the Innovation Transferability Platform including the Digital Twin based on the outcomes of Task 3.1.
- Chapter 4 presents the DT technical architecture and infrastructure, detailing all underlying components. It also describes the Model Library and the Execution Engine, which is responsible for executing model flows.
- Chapter 5 introduces the CitIQore application, designed and developed within T3.6, as a domain-specific, user-friendly layer on top of the DT platform.
- Chapter 6 is a non-technical chapter, positioning the Innovation Transferability Platform both within the URBANE project context and in the broader context of last mile logistics operations and Physical Internet (PI) implementation.
- Chapter 7 concludes the report by summarising the main takeaways and key results.

2 Requirements

This section introduces the functional and non-functional requirements that define what the system should do and how it should perform. Functional requirements describe the specific features and capabilities the platform must deliver, while non-functional requirements define the quality attributes such as performance, scalability, and security. Both categories are explained in more detail below. These requirements are designed to ensure that the Digital Twin Platform can support complex logistics simulations, integrate seamlessly with other URBANE tools, and operate reliably in diverse urban contexts.

2.1 Functional Requirements

The following requirements were identified through discussions with LL partners and insights gathered during workshops with project stakeholders. They reflect a detailed understanding of the solution's technical needs, shaped through close co-creation with end users to ensure that the platform meets practical operational demands.

TABLE 2. FUNCTIONAL REQUIREMENTS

ID	Requirement	Description
FR1	**Data Ingestion**	The system should be able to ingest data in real time or in batch from various external sources such as IoT devices, databases, and systems.
FR2	**Data Interoperability**	The system should semantically annotate models and metadata using Knowledge Graphs.
FR3	**Visualisation**	The system should allow the representation of both historical and real-time data with graphs, charts, 3D models or simulations.
FR4	**Analytics**	The system should generate performance metrics, key performance indicators (KPIs), and reports to enable decision-making and performance evaluation.
FR5	**Decision support**	The system should offer decision support capabilities using modelling capabilities in last mile logistics.
FR6	**Model integration**	The system should integrate a variety of ABM and AI models using a specified metadata configuration.
FR7	**Model Orchestration**	The system should allow the execution of chains of models, allowing them to be executed one after the other using the outputs of the former as inputs to the latter.
FR8	**Model Library**	The system should offer a library of reusable and transferrable AI and ABM models.

2.2 Non-Functional Requirements

Non-functional requirements are constraints on the services or functions offered by the system and apply to the system as a whole rather than individual system features or services. The following non-functional requirements for the URBANE Digital Twin Platform have been defined to ensure that the system meets essential performance, security, and usability standards. They set clear expectations for how the platform should operate under various conditions and align with best practices for delivering robust and reliable digital solutions (Table 3).

TABLE 3. NON-FUNCTIONAL REQUIREMENTS

Requirement	ID	Description
Security	NF1	System access must be restricted to authenticated and authorised users only (NF1).
	NF2	Only a subset of the authorised users is permitted to modify the system parameters.
Performance	NF3	All UI pages should load and render in under five seconds, for fifty concurrent users.
	NF4	The system should be capable of handling at least 100 different incoming data streams.
Scalability	NF5	The system should be able to cope with an increased number of connected users, but also an increasing size of datasets and system complexity without any performance degradation.
Availability	NF6	Users must be able to access their models and data assets at any moment with 99% reliability.
Usability & User Experience	NF7	The system should have an intuitive and user-friendly User Interface.
	NF8	It should offer comprehensive data visualisations and metrics.
Interoperability	NF9	The system should allow interoperability between different systems.

2.3 Model Integration Requirements

As part of WP3 tasks, several models have been developed including agent-based models, vehicle routing algorithms, CO_2 emissions calculation, demand estimation etc. all designed to help stakeholders analyse and optimise key aspects of last-mile logistics. The complete list of those models is presented in Table 4.

All models are developed in Python, which requires the hosting environment to prioritise the execution of Python code. To enable smooth and dynamic execution of diverse models, however, the platform is designed to manage detailed specifications for each model's semantics and metadata. This ensures that inputs, outputs, execution conditions, and resource requirements are clearly defined, allowing different models to be deployed, executed, and integrated consistently within the platform's overall architecture. Further details on these specifications are provided in Section 4.6.2.

In addition, the platform's architecture supports secure connections and API-based integrations, making it possible to incorporate externally hosted or closed-source models through standardised connectors and protocols whenever required. For instance, the CERTH model is integrated within the platform via a connector/API client, that is responsible to call the model that can be found on a web server on CERTH's premises and is only available for consumption via this REST API. Finally, models stemming from previous projects (i.e., the LEAD project) are already integrated in the DT platform and are available for further use and expansion. More details on the models are provided in Section 4.6.1.

TABLE 4. URBANE MODELS

Model	WP3 Task Number / Responsible Partner	Development Environment	Availability
HUMAT	T3.4 / NORCE	Python	Open within the consortium
Mass-GT	T3.4 / TUD	Python	Open source
Predictive Demand Modelling	T3.5 / SKEMA	Python	Open source
Dynamic Routing Model	T3.5 / SKEMA	Python	Open source
Dynamic Service Pricing and Booking	T3.5 / SKEMA	Python	Open source
Dynamic parcel reshuffling model	T3.5/ VLTN	Python	Open within the consortium
Parcel locker determination model	LL / CERTH	Python	Closed source (available via REST API)
UDR	T3.5/ INLE	Python	Open source

2.4 URBANE Living Lab Feedback and Additional Requirements

Following several interactions with the LLs through dedicated online meetings and on-site workshops (e.g., the Bologna Workshop 2023), new requirements emerged to improve the platform's usability. In addition, once the URBANE Digital Twin was made available to the Work Package 3 partners and modellers, their feedback led to further refinements. These updates focused on improving ease of use, particularly for non-modellers, by making the platform's features and tools more accessible and intuitive. The resulting requirements are summarised in Table 5.

TABLE 5. WP3 REQUIREMENTS

ID	Requirement	Description
FR11	**Domain specific**	The system should offer a domain-specific interface allowing for a more user-friendly experience.
FR22	**Data Visualisation**	The system should offer rich visualisation widgets.
FR33	**Output Calculations**	The system should allow for basic calculation capabilities (e.g., sum, avg..) to the models' results.

3 URBANE Conceptual Architecture

The European Commission has coined the term Local Digital Twin to differentiate Digital Twins representing cities and/or communities rather than traditional Digital Twins that represent industrial systems. LDTs are defined as *the virtual representation of the physical assets of a city or community, as well as the processes and systems connected to all the related data including data from the surrounding environment. "They use AI algorithms, data analytics and machine learning to create digital simulation models that can be updated and changed as their physical equivalents change. Real time, near real-time and historical data can be used in various combinations to provide the necessary capabilities for data analytics (descriptive, prescriptive, predictive), simulations and what-if scenarios"* [1]. In URBANE, the work undertaken in WP3 focuses on the design and development of a Local Digital Twin to support logistics operations with a special attention to last mile applied to cities of the project's Living Labs.

3.1 Design Steps & Objectives

Figure 3-1 presents the initial plan for the URBANE Innovation and Transferability Platform architecture design process. The process includes several iterations in the architecture design of the platform based on incoming specifications from the Living Labs (LLs) and digital models and a final validation against the requirements. The design process has been influenced by the work undertaken in other URBANE work packages and reports, such as D1.1 *URBANE framework for optimised green last mile operations*, the scoping documents of the Living Labs and the data request templates that were filled by the Living Labs and set the basis for the data collection process. The architecture design has been a collaborative process between WP3 partners; thus, the design process has been frequently updated through the participation in WP3 biweekly meetings as well as dedicated workshops and discussions with the different Living Lab partners.

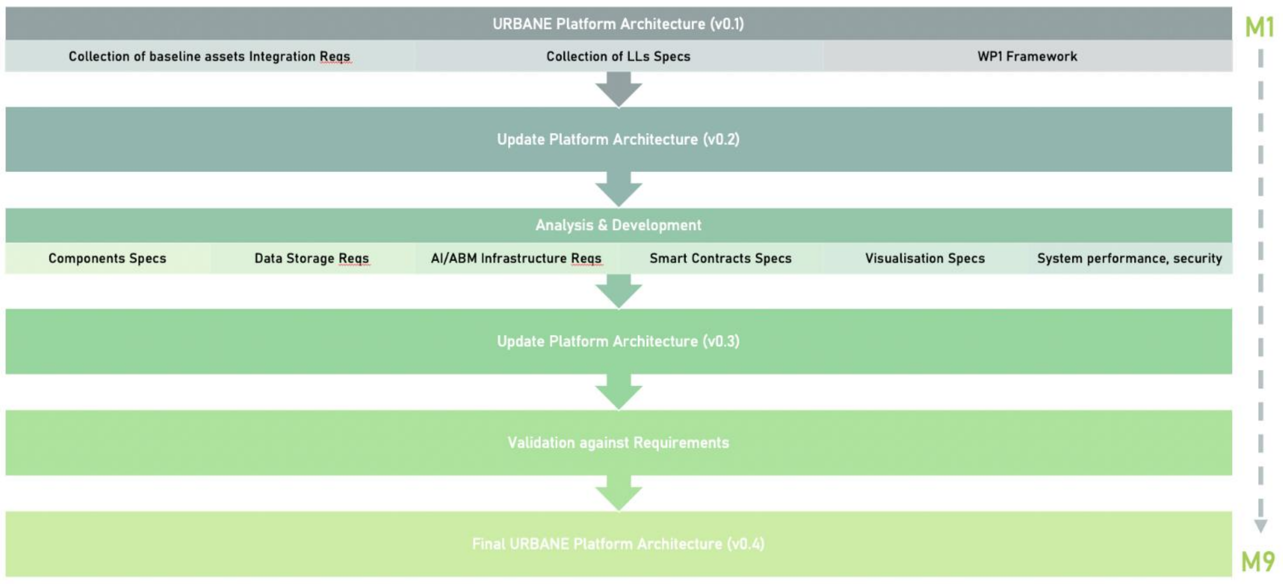

URBANE Platform Architecture (v0.1)
Collection of baseline assets Integration Reqs
Collection of LLs Specs
WP1 Framework
Update Platform Architecture (v0.2)
Analysis & Development
Components Specs
Data Storage Reqs
AI/ABM Infrastructure Reqs
Smart Contracts Specs
Visualisation Specs
System performance, security
Update Platform Architecture (v0.3)
Validation against Requirements
Final URBANE Platform Architecture (v0.4)
M1
M9

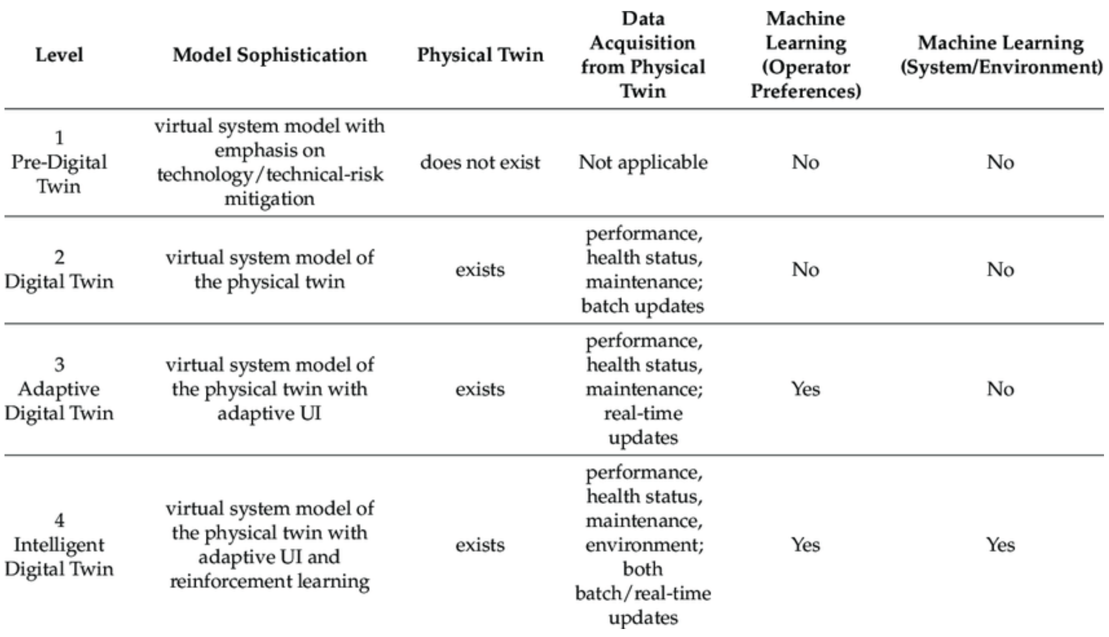
FIGURE 3-1. URBANE PLATFORM ARCHITECTURE PROCESS

3.2 Digital Twin Conceptual Architecture

Madnit et al. [2] defined four levels of virtual representations that reflect the maturity of a Digital Twin, as depicted in Figure 3-2. Within URBANE, the objective was to develop an Adaptive Digital Twin by acquiring both real-time and batch data, leveraging machine learning (ML) and other modelling techniques to represent its Physical Twin through an adaptive User Interface (UI).

The following sections describe the conceptual architecture of the URBANE Digital Twin (DT), structured according to the framework of the *Digital Twin Capabilities Periodic Table*. Subsequently, the software architecture is presented, with a particular focus

on how the different components of the URBANE Innovation Transferability Platform interact with each other, illustrated through the development of the relevant C4 diagrams[1].



Level	Model Sophistication	Physical Twin	Data Acquisition from Physical Twin	Machine Learning (Operator Preferences)	Machine Learning (System/Environment)
1 Pre-Digital Twin	virtual system model with emphasis on technology/technical-risk mitigation	does not exist	Not applicable	No	No
2 Digital Twin	virtual system model of the physical twin	exists	performance, health status, maintenance; batch updates	No	No
3 Adaptive Digital Twin	virtual system model of the physical twin with adaptive UI	exists	performance, health status, maintenance; real-time updates	Yes	No
4 Intelligent Digital Twin	virtual system model of the physical twin with adaptive UI and reinforcement learning	exists	performance, health status, maintenance, environment; both batch/real-time updates	Yes	Yes

FIGURE 3-2. DIGITAL TWIN MATURITY LEVELS

3.2.1 The Digital Consortium Periodic Table

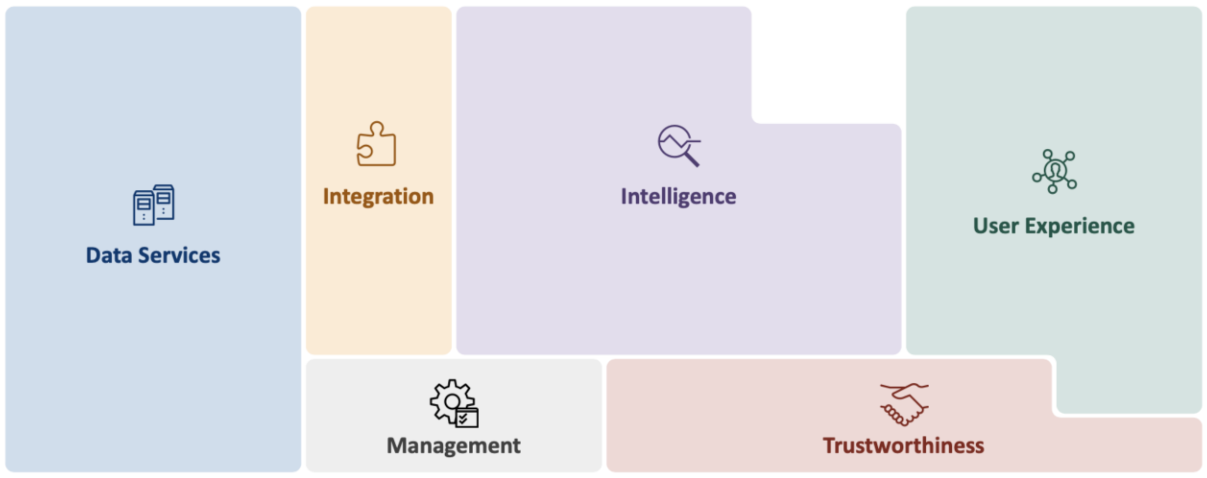


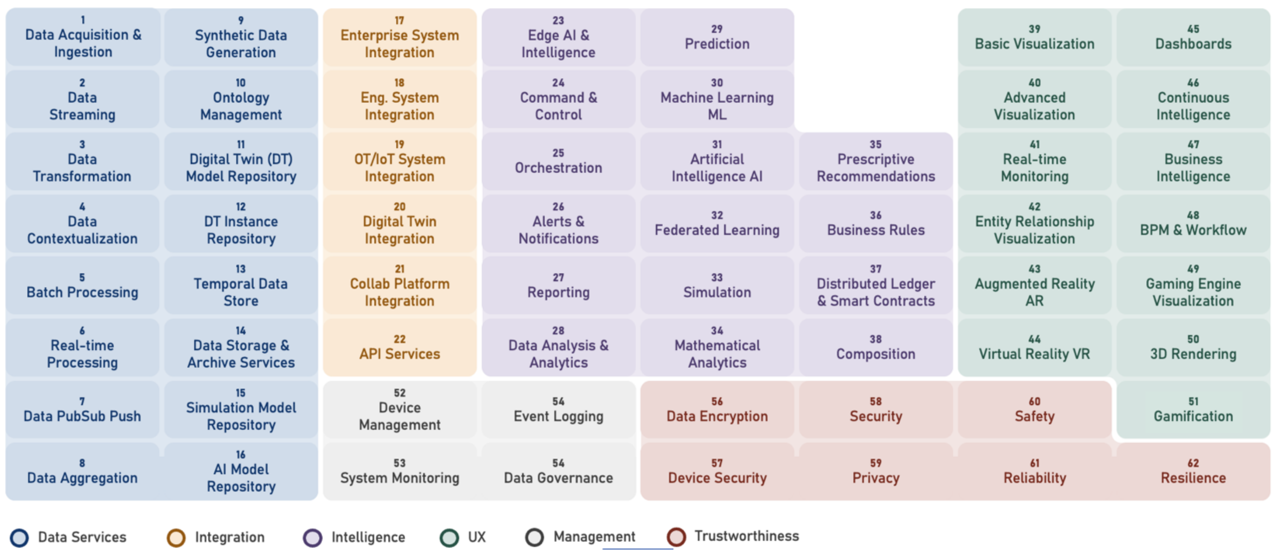
FIGURE 3-3. PERIODIC TABLE GROUPS OF CAPABILITIES

The Digital Twin Capabilities Periodic Table is a requirements-definition framework that is architecture and technology agnostic, developed by the Digital Twin Consortium. It aims at supporting multidisciplinary teams to design and develop Digital Twins in complex and large-scale environments. Following the Periodic Table approach, it categorises the different services that need to be offered by a DT into different groups of *capabilities*, as presented in Figure 3-3 and described in Table 6.

TABLE 6. CAPABILITIES PERIODIC SUB TABLE

Capability group	Description
Data Services	Provides the data pipeline of the DT from edge to data centre.
Integration	Provides access to internal & external systems and platforms. Interfacing between different DTs is also possible.
Management	Management and monitoring of system.
Intelligence	Intelligence services developed and offered on top of the data services. Analytics, AI, simulations & Blockchain solutions are among the intelligence capabilities.
User Experience	Data visualisation & User Interaction with the DT.
Trustworthiness	Safety, security, and privacy considerations.

Figure 3-4 presents the *Digital Twin Capabilities Periodic Table* consisting of 62 different capabilities that can be used to gather DT requirements for the different services required to develop a specific Use Case. Depending on the Use Case different combination of these capabilities may be considered necessary to address, while others can be ignored.



1 Data Acquisition & Ingestion	9 Synthetic Data Generation	17 Enterprise System Integration	23 Edge AI & Intelligence	29 Prediction		39 Basic Visualization	45 Dashboards
2 Data Streaming	10 Ontology Management	18 Eng. System Integration	24 Command & Control	30 Machine Learning ML		40 Advanced Visualization	46 Continuous Intelligence
3 Data Transformation	11 Digital Twin (DT) Model Repository	19 OT/IoT System Integration	25 Orchestration	31 Artificial Intelligence AI	35 Prescriptive Recommendations	41 Real-time Monitoring	47 Business Intelligence
4 Data Contextualization	12 DT Instance Repository	20 Digital Twin Integration	26 Alerts & Notifications	32 Federated Learning	36 Business Rules	42 Entity Relationship Visualization	48 BPM & Workflow
5 Batch Processing	13 Temporal Data Store	21 Collab Platform Integration	27 Reporting	33 Simulation	37 Distributed Ledger & Smart Contracts	43 Augmented Reality AR	49 Gaming Engine Visualization
6 Real-time Processing	14 Data Storage & Archive Services	22 API Services	28 Data Analysis & Analytics	34 Mathematical Analytics	38 Composition	44 Virtual Reality VR	50 3D Rendering
7 Data PubSub Push	15 Simulation Model Repository	52 Device Management	54 Event Logging	56 Data Encryption	58 Security	60 Safety	51 Gamification
8 Data Aggregation	16 AI Model Repository	53 System Monitoring	54 Data Governance	57 Device Security	59 Privacy	61 Reliability	62 Resilience

○ Data Services ○ Integration ○ Intelligence ○ UX ○ Management ○ Trustworthiness

FIGURE 3-4. THE CAPABILITIES PERIODIC TABLE (CPT)

[1] **C4 diagrams** are a visual modelling framework used to describe a software system's static structure and interactions at multiple levels of detail, from high-level context down to detailed components and code views (Brown, S. (2018). *Visualising software architecture with the C4 model*. Retrieved from https://c4model.com).

3.2.2 The URBANE Digital Twin

In accordance with the Capabilities Periodic Table, the URBANE Components diagram was developed, as shown in Figure 3-5 and Figure 3-6. The colouring used in both diagrams is based on the colouring used to group together the different capabilities of the Periodic Table. The Components diagram takes a closer look at each DT component by setting out the most critical features for the development of the URBANE DT. This is expected to be refined in the coming months in accordance with the LL requirements. For instance, in terms of storage it is expected that relational and non-relational databases will be required, but also pending on the LL requirements time-series databases may also be necessary.

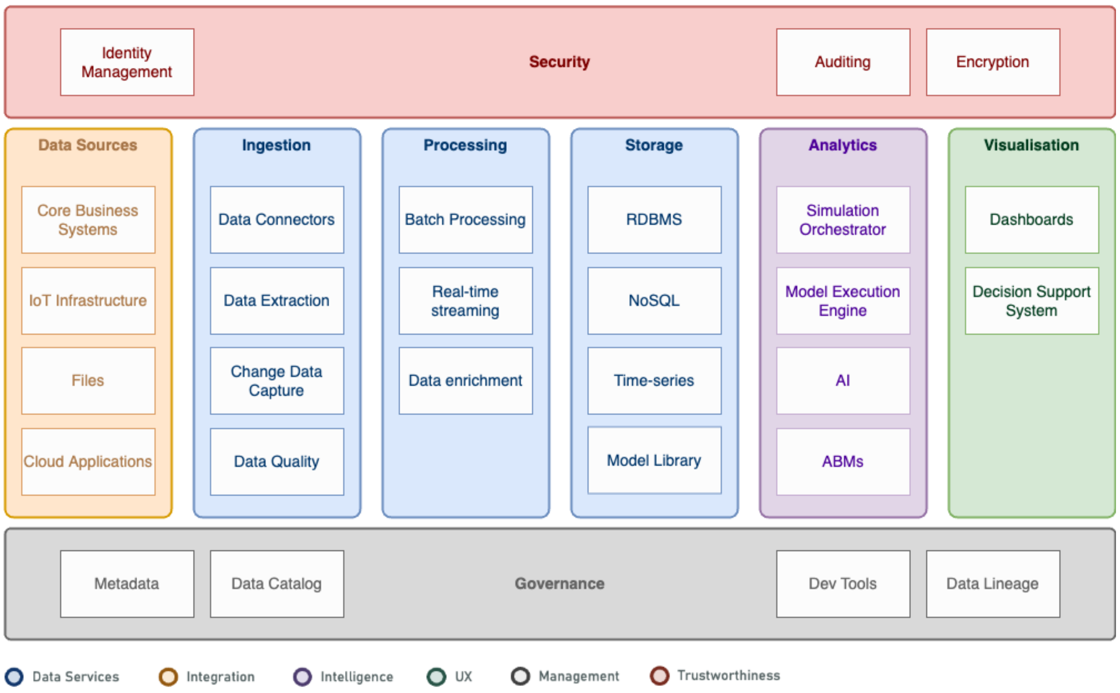


FIGURE 3-5. URBANE COMPONENTS DIAGRAM

The components of the URBANE platform are discussed in the following subsections.

Security – Trustworthiness

This group of components focuses on ensuring the security and trustworthiness of the data platform. It includes:

- **Identity Management:** Handles authentication and access control to ensure only authorized users can access the platform.
- **Auditing:** Tracks and monitors system activity to detect and investigate security incidents.
- **Encryption:** Protects data at rest and in transit by encrypting it to prevent unauthorised access.

Data Sources – Integration

This component handles the integration of various data sources into the platform. The goal is to gather data from various sources and make it available for processing and analysis. It includes but is not limited to the following sources:

- **Core Business Systems:** Integrates data from the organization's core business systems, such as ERP or CRM systems.
- **IoT Infrastructure:** Connects and integrates data from Internet of Things devices and sensors.

- **Files**: Ingests and integrates data from files in various formats, such as CSV, JSON, or XML.
- **Cloud Applications**: Integrates data from cloud-based applications and services.

Ingestion - Data Services

This component deals with the ingestion of data into the platform. It includes:

- **Data Connectors**: Establishes connections and protocols to retrieve data from various sources and formats.
- **Data Extraction**: Extracts data from the sources and transforms it into a suitable format for ingestion.
- **Change Data Capture**: Tracks and captures only the changes made to the data sources, minimizing unnecessary data transfer.
- **Data Quality**: Ensures the reliability and accuracy of the ingested data through validation, cleansing, and standardization processes.

Processing - Data Services

This component handles the processing of data within the platform. It includes batch processing for large-scale data processing, real-time streaming for processing data in real-time, and data enrichment to enhance and augment the data with additional information. In more detail:

- **Batch Processing**: Performs large-scale data processing in scheduled batches for efficient data analysis.
- **Real-time Streaming**: Processes data in real-time as it is generated, enabling immediate insights and responses.
- **Data Enrichment**: Enhances and augments the data with additional information or derived attributes to provide more meaningful insights.

Storage - Data Services

This component provides storage capabilities for the data platform. It includes various types of storage options such as relational databases (RDBMS), NoSQL databases, time-series databases, and a model library to store and manage machine learning and other types of models.

Analytics – Intelligence

This component focuses on analysing and deriving insights from the data. It includes a simulation orchestrator for running simulations and scenario analysis, a model execution engine for executing machine learning models, artificial intelligence capabilities for advanced analytics, and agent-based models (ABMs) for simulating complex systems. In more detail, the sub-components are:

- **Simulation Orchestrator**: Coordinates and manages simulations and scenario analysis to predict outcomes and understand system behaviour.
- **Model Execution Engine**: Executes various types of models and model scenarios to analyse data and generate predictions or classifications.
- **Artificial Intelligence (AI)**: Applies AI models and techniques to uncover patterns, insights, and automate decision-making processes.
- **Agent-Based Models (ABMs)**: Simulates complex systems by modelling individual agents' behaviour and interactions.

Visualisation – UX

This component handles the visualization and user experience aspects of the data platform. It includes the creation of interactive dashboards for data visualization and exploration, as well as decision support systems to assist users in making data-driven decisions.

- **Dashboards:** Present data and insights in visually appealing and interactive dashboards for easy consumption and exploration.
- **Decision Support Systems**: Provides tools and interfaces that assist users in making data-driven decisions effectively and efficiently.

Governance – Management

This component is responsible for the management of the data platform. It includes metadata management to organise and describe the data assets, a data catalogue to provide a centralized view of available data, development tools for building and managing the platform, and data lineage to track the origin and history of data within the platform.

- **Metadata**: Manages information about data assets, such as definitions, structures, and relationships, to enable effective data management and understanding.
- **Data Catalogue**: Provides a centralised inventory and search interface for all available data assets within the platform.
- **Dev Tools**: Offers development tools and environments to support the creation, testing, and deployment of data platform components.
- **Data Lineage**: Tracks and documents the origin, transformation, and movement of data throughout the platform, ensuring traceability and compliance.

Finally, the following diagram takes a step further by pinpointing which URBANE task and subtask is responsible for the development of each component (Figure 3-6).

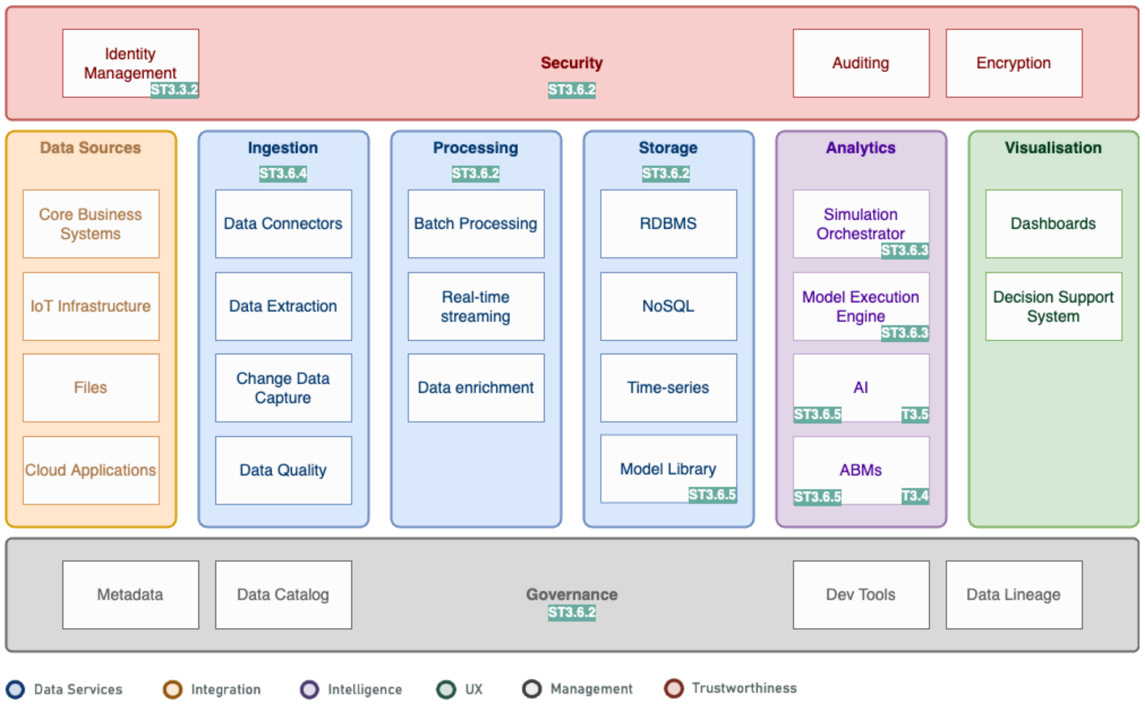


FIGURE 3-6. URBANE COMPONENTS DIAGRAM ANNOTATED WITH PROJECT TASKS

3.3 Innovation Transferability Platform Software Architecture

This section presents the software architecture of the URBANE platform that was developed in the process of understanding and communicating in more depth the architecture behind the URBANE Innovation Transferability Platform and goes one level lower than the conceptual architecture.

The C4 (Context, Containers, Components, and Classes) diagram technique was used to create the software architecture of the URBANE platform. The C4 technique is a hierarchical approach to visualize the architecture of a software system. It provides a clear and concise way to depict the different levels of abstraction in a system, from high-level context to low-level implementation details. The C4 diagram technique consists of four diagram types:

1. **Context Diagram**: Illustrates the high-level system boundaries and its interactions with external entities, such as users, systems, and external services.
2. **Container Diagram**: Focuses on the internal components or containers that make up the system and shows their relationships, interactions, and dependencies.
3. **Component Diagram**: Breaks down individual containers into their respective components or modules, showing the internal structure and dependencies within each container.
4. **Class Diagram**: Represents the detailed structure and relationships between classes within a component, showcasing the implementation details and associations.

The C4 diagram technique helps architects and developers communicate the system's architecture effectively, enabling discussions around system behaviour, dependencies, and design decisions at different levels of granularity. It provides a scalable and modular approach to visualize and understand complex software systems. In URBANE were utilised the first 3 diagram types as the Class Diagram was too detailed for our purposes, but it can be used during the step of the actual implementation of the platform.

The purpose behind the C4 diagrams presented in this section is to depict the architecture of the WP3 components as a whole but also at different levels of abstraction.

3.3.2 Context Diagram

The first diagram developed is the Context Diagram, as displayed in Figure 3-7. The Innovation Transferability Platform is placed at the centre of this diagram, while it is surrounded by other LL systems it may interact with, external data sources that enable data ingestion into the platform, as well as the users of the Platform. At this stage of the project three main user roles have been identified:

- Modeller: A project partner who is owner of one or more URBANE models. They wish to develop and integrate their model into the DT Platform. Overall, modellers have a scientific background in an engineering discipline related to Transport and Logistics (T&L).
- Decision-making Consultant: A stakeholder wishing to create and simulate different scenarios utilizing and/or combining the existing models using real life data for better decision-making. In URBANE that would be LL stakeholders wishing to answer different "What-if" questions through the relevant models. They do not need to have a technical or engineering background; they are rather domain experts that use the DT to perform data-driven decision making by acquiring information related to last-mile logistics interventions and so on.

- Operations Manager/Consultant: A stakeholder who wishes to observe the current state of things in the physical world and view KPI predictions. They are not expected to have a technical background; they make use of the DT to view the logistics network at an operational level.

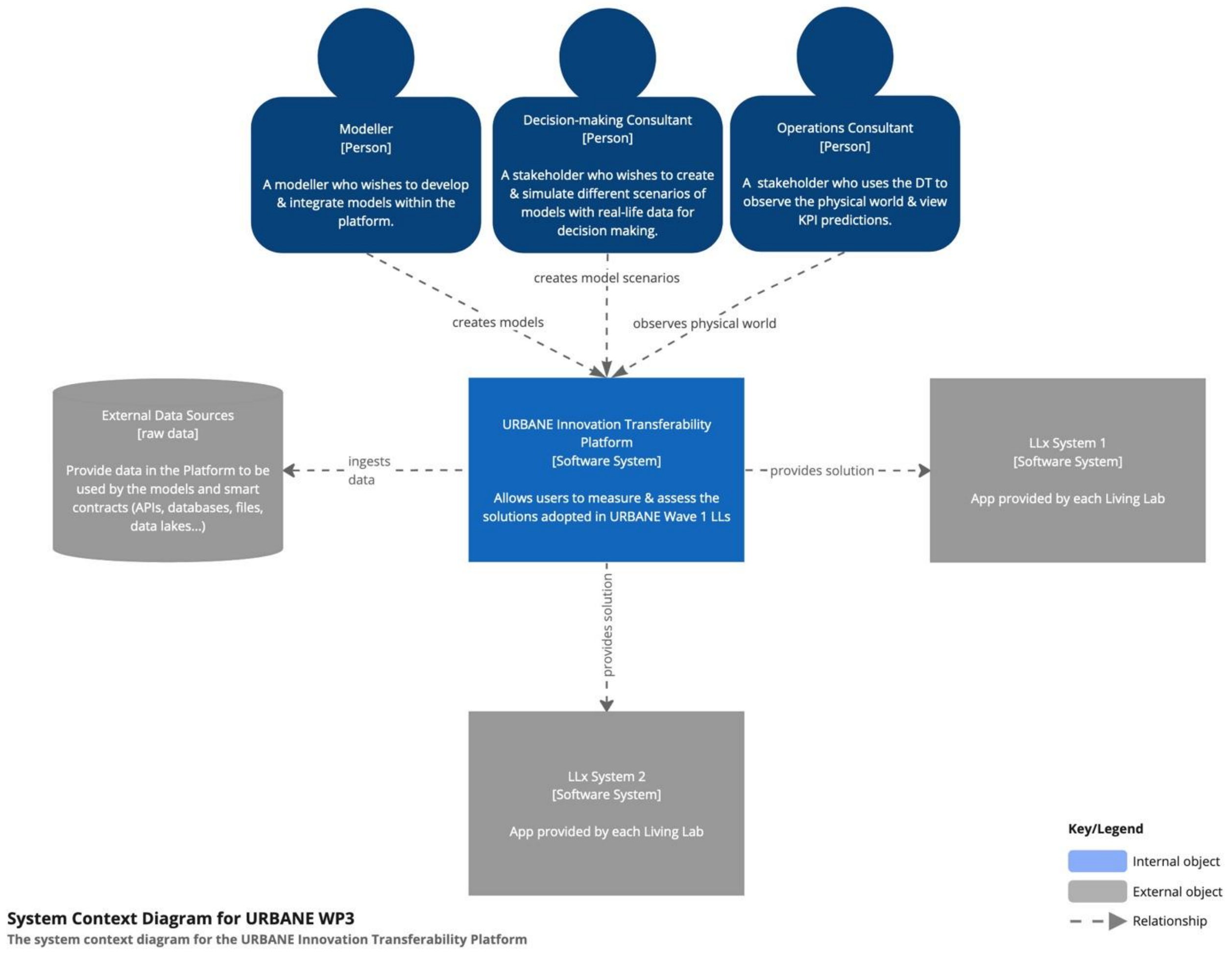

Modeller
[Person]
A modeller who wishes to develop & integrate models within the platform.
Decision-making Consultant
[Person]
A stakeholder who wishes to create & simulate different scenarios of models with real-life data for decision making.
Operations Consultant
[Person]
A stakeholder who uses the DT to observe the physical world & view KPI predictions.
creates model scenarios
creates models
observes physical world
External Data Sources
[raw data]
Provide data in the Platform to be used by the models and smart contracts (APIs, databases, files, data lakes...)
ingests data
URBANE Innovation Transferability Platform
[Software System]
Allows users to measure & assess the solutions adopted in URBANE Wave 1 LLs
provides solution
LLx System 1
[Software System]
App provided by each Living Lab
provides solution
LLx System 2
[Software System]
App provided by each Living Lab
Key/Legend
Internal object
External object
Relationship
System Context Diagram for URBANE WP3
The system context diagram for the URBANE Innovation Transferability Platform

FIGURE 3-7. CONTEXT DIAGRAM. INNOVATION TRANSFERABILITY PLATFORM

3.3.3 Container Diagram

The Container Diagram presented in Figure 3-8 takes a closer look at the Innovation Transferability Platform by pinpointing its main services/containers, which are included in the technical tasks of WP3. These are: (a) the URBANE Digital Twin, (b) the Blockchain Service, (c) the Data-Driven Impact Assessment Radar, and finally the modelling containers, i.e., (d) the Agent Based Models (ABMs) and the (e) AI models. The diagram also illustrates how the different users interact with the WP3 services/containers.

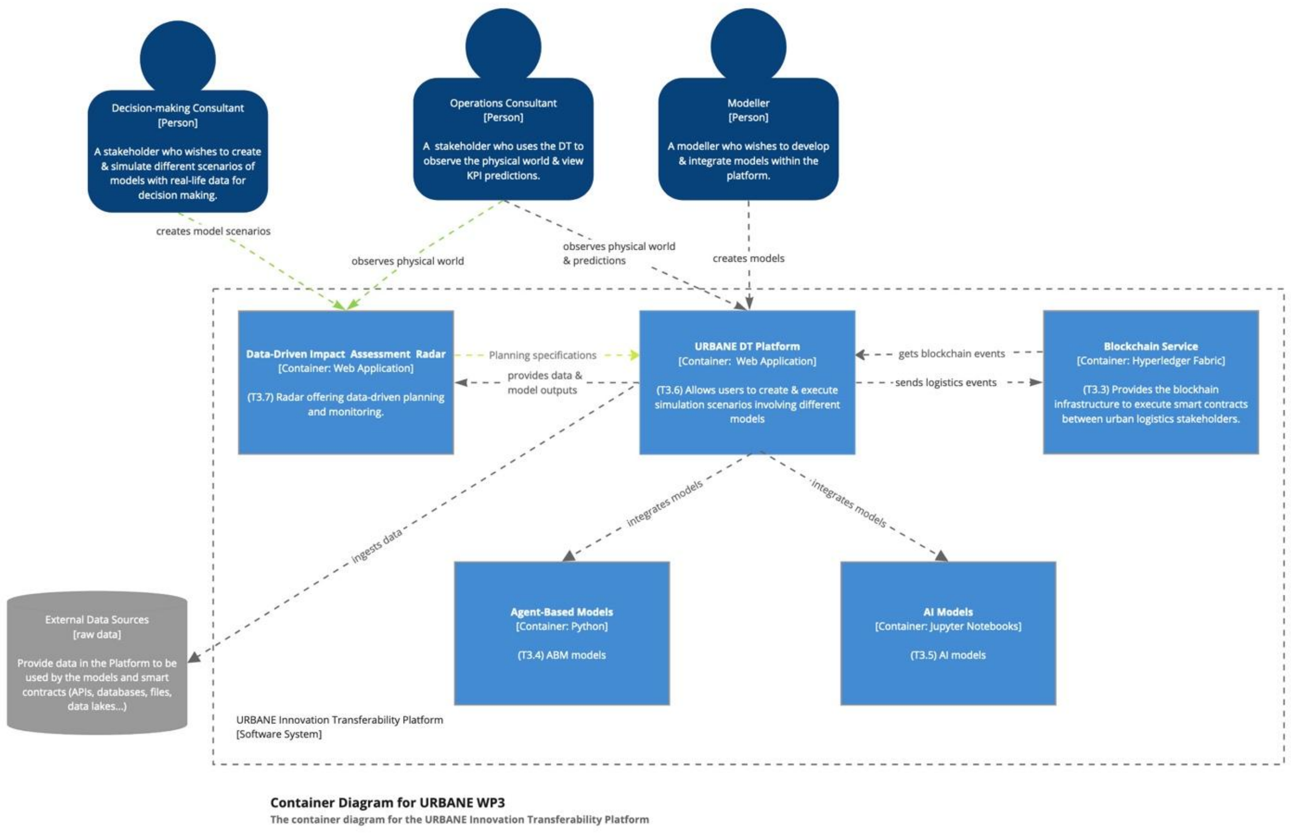

Decision-making Consultant
[Person]
A stakeholder who wishes to create & simulate different scenarios of models with real-life data for decision making.
Operations Consultant
[Person]
A stakeholder who uses the DT to observe the physical world & view KPI predictions.
Modeller
[Person]
A modeller who wishes to develop & integrate models within the platform.
creates model scenarios
observes physical world
observes physical world & predictions
creates models
Data-Driven Impact Assessment Radar
[Container: Web Application]
(T3.7) Radar offering data-driven planning and monitoring.
Planning specifications
provides data & model outputs
URBANE DT Platform
[Container: Web Application]
(T3.6) Allows users to create & execute simulation scenarios involving different models
gets blockchain events
sends logistics events
Blockchain Service
[Container: Hyperledger Fabric]
(T3.3) Provides the blockchain infrastructure to execute smart contracts between urban logistics stakeholders.
ingests data
integrates models
integrates models
External Data Sources
[raw data]
Provide data in the Platform to be used by the models and smart contracts (APIs, databases, files, data lakes...)
Agent-Based Models
[Container: Python]
(T3.4) ABM models
AI Models
[Container: Jupyter Notebooks]
(T3.5) AI models
URBANE Innovation Transferability Platform
[Software System]
Container Diagram for URBANE WP3
The container diagram for the URBANE Innovation Transferability Platform

FIGURE 3-8. CONTAINER DIAGRAM. INNOVATION TRANSFERABILITY PLATFORM

3.3.4 Component Diagrams

Following the design of the previous two diagrams, the WP3 partners were asked to develop Component Diagrams for the three main building blocks of WP3.

The Component Diagram for the DT Platform can be viewed in Figure 3-9. It highlights the components of its underlying architecture and how all these are brought together to offer the DT functionalities to the users through the DT Portal. The key components are:

- Data Connectors that are responsible for the ingestion of external data into the DT.
- An Event System, i.e., an event streaming service that processes incoming events (Apache Kafka or similar).
- The Data Processing Engine that performs data transformations and processing (Apache Spark or similar).
- A Knowledge Graph providing a semantic network representation of the impact assessment data for LLs and the corresponding model metadata that are integrated into the DT.
- Data Storage. An architecture using a plethora of storage solutions depending on the DT needs that collects and store all the system data.
- Data Catalogue. An organised inventory of metadata describing all the data of the DT, thus enabling data discovery and governance (Datahub, Apache Atlas etc.).
- A Rules Engine allowing the employment of business logic within the DT.
- Model Orchestration enabling the execution of chains of models (Apache Airflow).
- Finally, the Model Library collects all the metadata related to the models integrated into the DT.

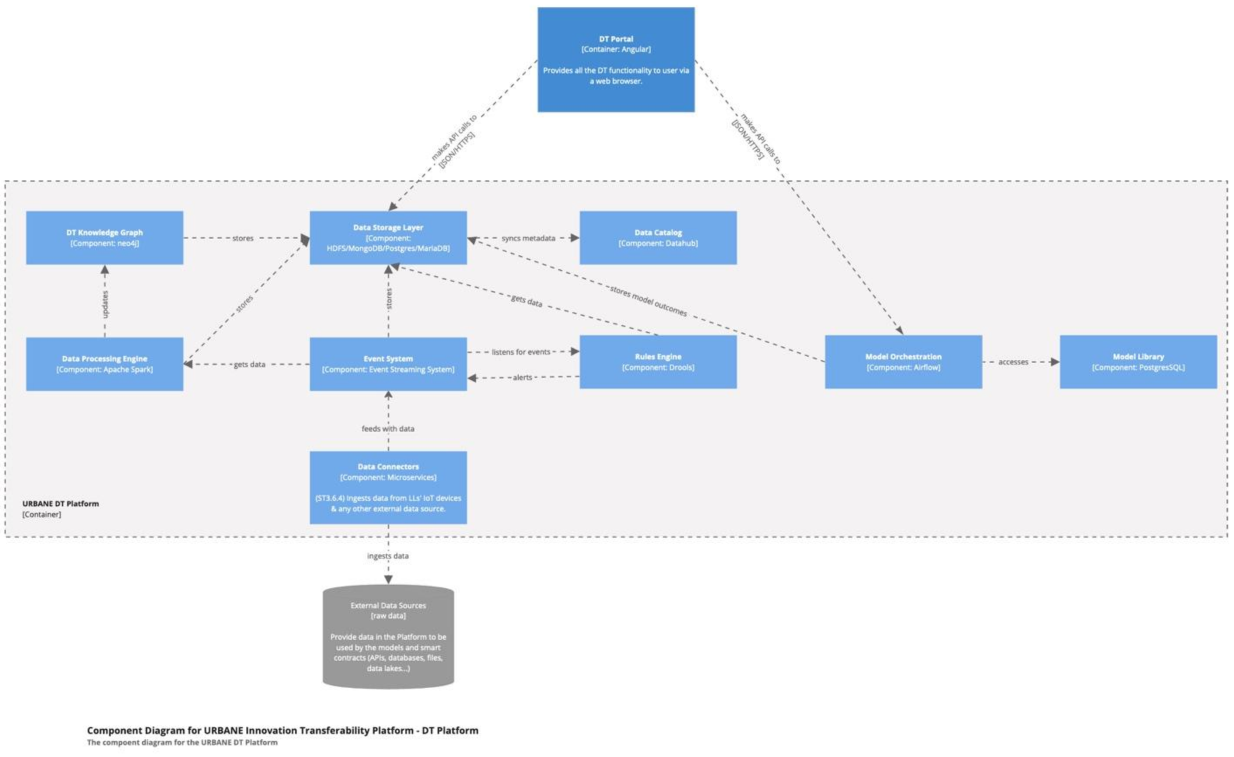

DT Portal
[Container: Angular]
Provides all the DT functionality to user via a web browser.
makes API calls to [JSON/HTTPS]
makes API calls to [JSON/HTTPS]
DT Knowledge Graph
[Component: neo4j]
stores
Data Storage Layer
[Component: HDFS/MongoDB/Postgres/MariaDB]
syncs metadata
Data Catalog
[Component: Datahub]
updates
stores
stores
gets data
stores model outcomes
Data Processing Engine
[Component: Apache Spark]
gets data
Event System
[Component: Event Streaming System]
listens for events
alerts
Rules Engine
[Component: Drools]
Model Orchestration
[Component: Airflow]
accesses
Model Library
[Component: PostgresSQL]
feeds with data
Data Connectors
[Component: Microservices]
(ST3.6.4) Ingests data from LLs' IoT devices & any other external data source.
URBANE DT Platform
[Container]
ingests data
External Data Sources
[raw data]
Provide data in the Platform to be used by the models and smart contracts (APIs, databases, files, data lakes...)
Component Diagram for URBANE Innovation Transferability Platform - DT Platform
The compoent diagram for the URBANE DT Platform

FIGURE 3-9. COMPONENT DIAGRAM. DT PLATFORM

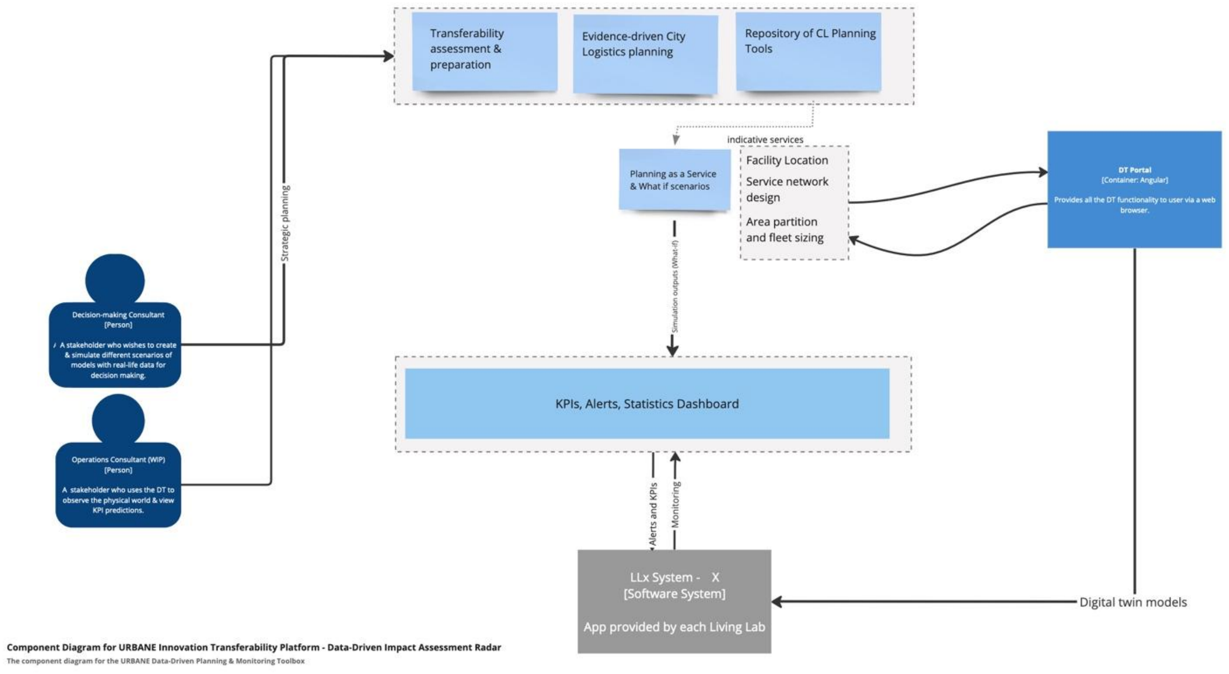

Transferability assessment & preparation
Evidence-driven City Logistics planning
Repository of CL Planning Tools
indicative services
Planning as a Service & What if scenarios
Facility Location
Service network design
Area partition and fleet sizing
DT Portal
[Container: Angular]
Provides all the DT functionality to user via a web browser.
Strategic planning
Decision-making Consultant
[Person]
A stakeholder who wishes to create & simulate different scenarios of models with real-life data for decision making.
Operations Consultant (WIP)
[Person]
A stakeholder who uses the DT to observe the physical world & view KPI predictions.
Simulation outputs (What-if)
KPIs, Alerts, Statistics Dashboard
Alerts and KPIs
Monitoring
LLx System - X
[Software System]
App provided by each Living Lab
Digital twin models
Component Diagram for URBANE Innovation Transferability Platform - Data-Driven Impact Assessment Radar
The component diagram for the URBANE Data-Driven Planning & Monitoring Toolbox

FIGURE 3-10. COMPONENT DIAGRAM. DATA-DRIVEN IMPACT ASSESSMENT RADAR

The second diagram represents the architecture behind the Data-Driven Impact Assessment Radar (Figure 3-10) and how it interacts with the URBANE Digital Twin. Further details into this component and the interaction between these two components are given in section 4.5.

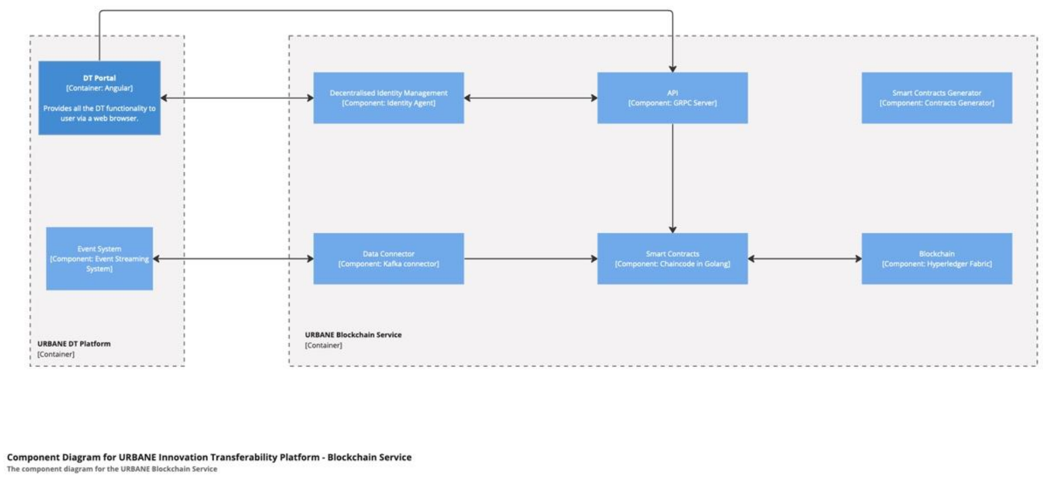

DT Portal
[Container: Angular]
Provides all the DT functionality to user via a web browser.
Event System
[Component: Event Streaming System]
URBANE DT Platform
[Container]
Decentralised Identity Management
[Component: Identity Agent]
API
[Component: GRPC Server]
Smart Contracts Generator
[Component: Contracts Generator]
Data Connector
[Component: Kafka connector]
Smart Contracts
[Component: Chaincode in Golang]
Blockchain
[Component: Hyperledger Fabric]
URBANE Blockchain Service
[Container]
Component Diagram for URBANE Innovation Transferability Platform - Blockchain Service
The component diagram for the URBANE Blockchain Service

FIGURE 3-11. COMPONENT DIAGRAM. BLOCKCHAIN SERVICE

The final component diagram represents the Blockchain Service (Figure 3-11) and its interaction with the URBANE Digital Twin Platform. The Blockchain Service is described in detail in D3.1 *Collaboration Governance Ledger & Smart Contracts*. Special attention, however, should be placed on how the Blockchain Service interacts with the URBANE DT. The first interaction occurs through the exchange of events between the two services (also evident in the Container diagram in Figure 3-8). The URBANE DT sends logistics events stemming from the LLs to the Blockchain, while the Blockchain returns Blockchain events (interactions occur via the dedicated streaming platform that is Apache Kafka[2]). The second interlink between the two occurs through the Decentralised Identity Management service that offers a distributed blockchain-based and privacy-preserving identity management scheme to the DT Platform. The detail description of the services and technical implementations are explained in the respective deliverable report submitted - D3.1 *Collaboration Governance Ledger & Smart Contracts*.

[2] https://kafka.apache.org/

4 Digital Twin Infrastructure

This chapter presents the infrastructure developed in the context of Task 3.6; the Digital Twin infrastructure including the modelling, simulation environment developed to allow for the integration and execution of the models developed in Tasks 3.4 and 3.5.

4.1 Digital Twin Technical Architecture

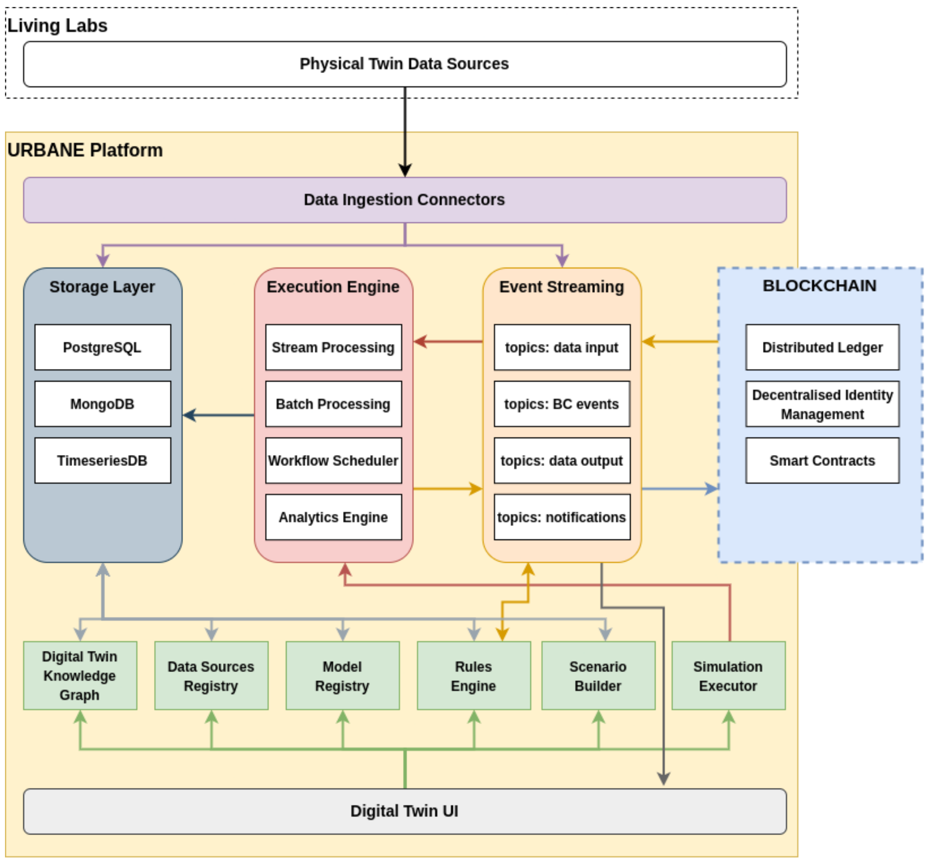


FIGURE 4-1. URBANE TECHNICAL ARCHITECTURE OF DIGITAL TWIN

The URBANE Digital Twin is a central service of the technological core of the project. Its technical architecture is shown in Figure 4-1. Data is ingested into the DT from different data sources offered by the Living Labs. Data Ingestion Connectors are used to ingest the data into the DT either in batch or in real time through event streaming. Following that, real-time data is handled as events via Apache Kafka and are assigned into "data input" Kafka topics. The Event Streaming service handles also events stemming from the Blockchain service, but also any notifications that may be generated from the system. The Execution Engine is responsible for data processing (stream, batch, but also analytics), while the storage layer supports different data purposes by offering relational, NoSQL, but also time series storage options.

Figure 4-2 presents the DT architecture from a different viewpoint; as a Big Data Platform focusing on the data pipeline, i.e., illustrating the movement and processing of data stemming from LL data sources all the way until they are stored in the platform's storage layer and exposed to the users through a dedicated User Interface. The Event Streaming Layer of Figure 4-1 is represented here by the Kafka topics currently in use by the LLs, the Execution Engine is shown as the Ingestion Services (Spark cluster), while the Storage Layer is depicted by the various storage solutions offered by the platform (i.e., mongoDB, MySQL, influxDB etc.). Finally, the Blockchain Service is represented by Hyperledger Fabric, while the Blockchain Backend Service responsible for the interactions between the DT's backend services, Dashboard (frontend) and Fabric.

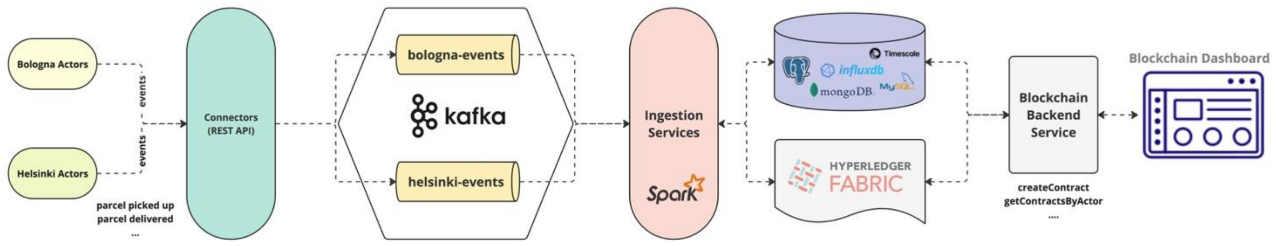


FIGURE 4-2. URBANE DATA PIPELINE

4.2 Open DT modelling and simulation environment

This section describes the essential configuration and technical foundations needed to deploy and operate the modelling and simulation environment of the URBANE Digital Twin successfully. It outlines the key components, integration steps, and supporting infrastructure that enable the platform to run diverse models, manage data flows, and deliver reliable results for last-mile logistics scenarios.

4.2.1 Model Integration Specifications

Models are executed one after another forming a chain of models or a scenario, as it is called in the URBANE DT. Figure 4-3 presents an example execution engine that illustrates how two models are containerised using Docker and are executed one after another using Apache Airflow. The Echelon/2-echelon[3] model is an open-source model developed in R, and it is executed first. Its outputs are then used as input parameters into the copert[4] model; a closed model that was integrated into the DT using API communication.

Evidently, for this process to run smoothly it is important that all the models have already been successfully integrated into the DT. That implies that each model needs to have proper configuration:

- they have clear input and output variables.
- The code inside each model does not contain hardcoded variables.
- Models can run successfully in any machine (not just the modeller's local machine).
- Hence, the need for containerisation of the models is necessary. As displayed in Figure 4-3 Docker containers are used to package each individual model.

[3] https://github.com/Horizon-LEAD/2echelon
[4] https://www.emisia.com/utilities/copert/

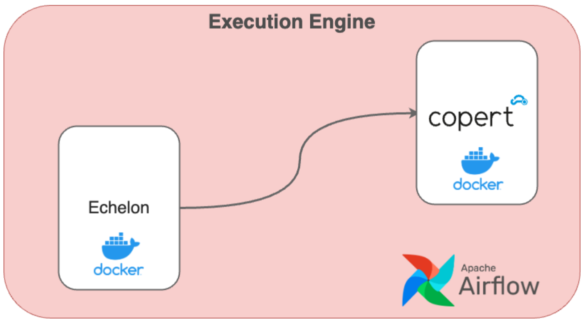
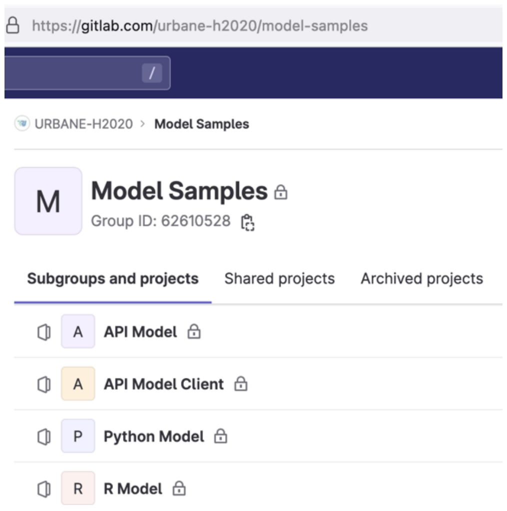


FIGURE 4-3. MODEL EXECUTION ENGINE

To that end, model samples were created and are available at the URBANE GitLab repository: https://gitlab.com/urbane-h2020/model-samples

All project partners involved in the development of models were encouraged to go through these templates and even use them in their modelling efforts, as they were specifically developed to smooth the model integration process into the DT. The model samples include guidelines (these can be found in the Annex) on how to set up the environment for a model, the execution of the model, and how to define and set up input and output data.

As depicted in Figure 4-4 several samples were developed depending on the language the modellers are using, including cases when the model code is closed; hence API communication is required for the models to be integrated.


https://gitlab.com/urbane-h2020/model-samples

URBANE-H2020 › Model Samples

Model Samples
Group ID: 62610528

- API Model
- API Model Client
- Python Model
- R Model

FIGURE 4-4. MODEL SAMPLES IN GITLAB

At this point it is worth highlighting that transferability is one of the key URBANE objectives. The technical solutions developed in WP3 were applied to the Lighthouse LLs but were also transferred to the Twinning LLs during the second phase of the project. Therefore, reusability and transferability of the developed models was from the beginning a matter of utmost importance during the project's lifecycle.

4.2.2 Model Containerisation Process

The previous section presented the required specifications for a smooth integration process of each model into the URBANE DT Platform. In this section we present the required process for the containerisation of the URBANE models, which is the one of the final steps presented in Annex I ("Containerise model" step). The containerisation was undertaken by the model integration team in collaboration with the modellers of WP3 (as part of the work in ST3.6.5).

As a first step, a Docker installation is expected to run on the local machine or the target environment where the containerised model shall be deployed. The process involves several technical steps which are described as follows:

- Creation of a *Dockerfile*: We begin by creating a *Dockerfile*, i.e., a text file containing instructions on how to construct a Docker image. This file delineates the base image, dependencies, and setup commands necessary for establishing the model's environment.
- Package Model Code: The model code needs to be set up accordingly, typically housed within a *src* folder, alongside any requisite files (Ensure the existence of a sample-data directory containing separate folders for input and output files) within the same directory as the *Dockerfile*. The *src* folder should comprise of the following:
    - *main.py*: Serves as the model's entry point, encompassing an argument parser for input files (*CLI*, *.env*) and defining logging procedures.
    - *init.py*: Specifies the version of the model.
    - *envctl.py*: Manages environmental parameters relevant to the model.
    - *proc.py*: Functions as the orchestrator for the model's operations.
    - *functions.py*: Contains various functions integral to the model.
- Generation of the *.gitignore* file: This file specifies which files are to be ignored by version control systems to prevent unnecessary tracking.
- Definition of dependencies: Creation of a *requirements.txt* file enumerating all Python dependencies indispensable for the given model. This file will be copied into the Docker container and used for dependency installation during the build phase.
- Creation of *setup.py*: This is a Python script facilitating the packaging of Python projects. This file encompasses metadata concerning the model, its dependencies, and the entry point.
- Generation of *modelspec.json*: This file is critical, as it enables the URBANE platform to correctly adapt the model. This is further explained in section 4.6.
- Adding model description in *README.md*: The *README.md* file offers a summary of the model's scope along with comprehensive instructions on how to build and execute the model.
- Building Docker Image: After launching a terminal or command prompt, we navigate to the directory housing the Dockerfile and model code and execute the build command to construct the Docker image.
- Running Docker Container: Following the successful Docker image construction, we can now instantiate a container based on the image.

4.3 Virtual Model Execution Engine – Simulations Orchestrator

The Model Execution Engine is responsible for the orchestration and execution of the URBANE models including the facilitation of the execution of standalone models or sequences of models – i.e., one after another, using the outputs of one model as inputs to the model that follows.

The URBANE models developed in the context of Tasks 3.4 and 3.5 - described in section 4.6.1 – were integrated in the URBANE Digital Twin following a Model Integration Process, explained in detail in section 4.6. Once the models are available in the platform, they can be used by the model execution engine for their execution on the platform. More specifically, from that moment on, the platform recognises the metadata of each model, notably their inputs, outputs, and execution parameters.

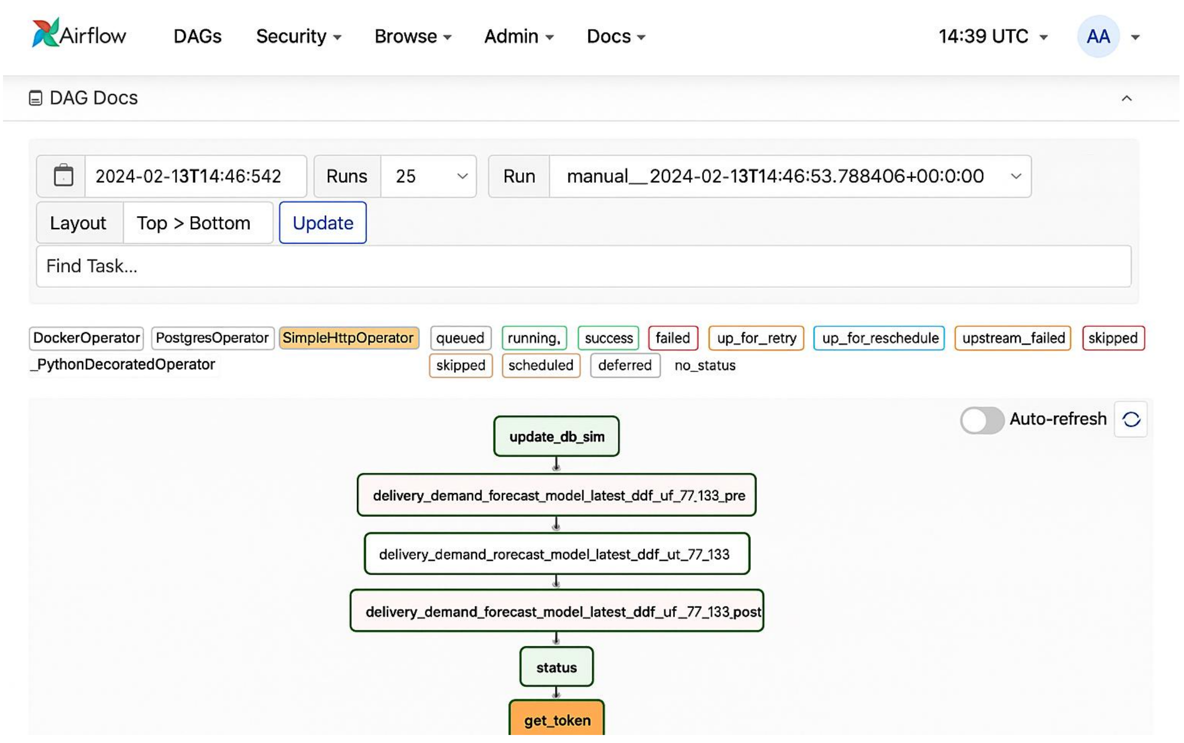


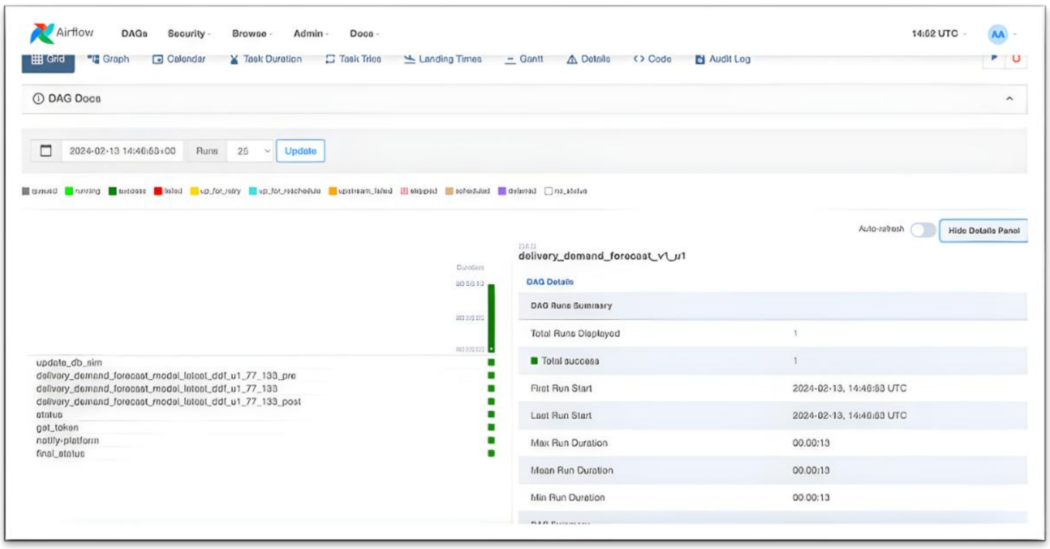
FIGURE 4-5. AIRFLOW SCHEDULER DAG EXAMPLE WITH T3.5 MODELS

At this point, it is worth identifying the three key entities that are offered in the URBANE DT and their differentiations:

1. *Model*. Models refer to the aforementioned Agent-Based and AI models developed by the project's modellers in Tasks 3.4 and 3.5. Once integrated in the Platform, they are treated as black boxes that contain code, whose outputs may change when different inputs are offered.
2. *Scenario*. This term is used when referring to the sequences of models that are created by the Platform users, allowing the models to be executed in a specific fashion (usually one after the other).
3. *Simulation.* The executions of the scenarios with specific input parameters are called simulations. Simulations can be triggered from the User Interface allowing the models to run in sequence and finally presenting the results of the simulations in dedicated Dashboards.

In the Platform's backend system, a specific component called Simulation Manager is in charge of retrieving the configuration of each simulation, validating the input parameters and passing the necessary information regarding the execution of the simulation to another component, known as the Simulation Orchestrator. The Orchestrator is responsible for the dynamic creation of simulation execution environments. Apache Airflow[5], a well-known open-source workflow management system, offers the foundation for the Simulation Orchestrator. Airflow employs Directed Acyclic Graphs (DAGs) for the workflow orchestration. DAGs identify the *tasks*, *workflows*, and their in-between *relationships*. In URBANE the *tasks* refer to the model executions, and each DAG represents an URBANE Scenario.

All models have specific execution and operational requirements, which are defined within the Model Library (section 4.6.1) and instantiated when needed for their executions. These environments are employed when needed and upon finalisation of the execution they are deleted for resources preservation. Due to the parameterisation of the models and the temporal requirement of execution cycles, multiple instances are dynamically created for each scenario evaluation.



delivery_demand_forecast_v1_u1

update_db_sim
delivery_demand_forecast_model_latest_ddf_u1_77_133_pre
delivery_demand_forecast_model_latest_ddf_u1_77_133
delivery_demand_forecast_model_latest_ddf_u1_77_133_post
status
get_token
notify-platform
final_status

DAG Runs Summary	
Total Runs Displayed	1
🟩 Total success	1
First Run Start	2024-02-13, 14:40:53 UTC
Last Run Start	2024-02-13, 14:40:53 UTC
Max Run Duration	00:00:13
Mean Run Duration	00:00:13
Min Run Duration	00:00:13

FIGURE 4-6 AIRFLOW: MODEL EXECUTION STATUS

A DAG in a given Scenario contains an initialisation and a finalisation task. The former updates the storage layer (PostgreSQL) that it pairs a simulation execution to a specific DAG run ID through a PostgresOperator[6]. The latter performs the extraction of any declared virtual output variable. The models are executed using a DockerOperator[7] that manages the execution of the related container. Finally, a task implementing a callback HTTP request (HttpOperator[8]) to the Platform notifies regarding the finalisation of the simulation and triggers relevant status updates. DAG runs can be triggered on demand by the Airflow user, but scheduled execution of DAGs is also supported. An Airflow DAG[9] is written in Python, always placed in a specific Airflow directory. Airflow Operators are used to define what is executed in the workflow. Numerous options are offered, such as BashOperator, PythonOperator, PostgresOperator, aspiring to fulfil any model specifications. Once a DAG is generated, dynamically user parameters and inputs are offered.

[5] https://airflow.apache.org/
[6] https://airflow.apache.org/docs/apache-airflow-providers-postgres/stable/operators/postgres_operator_howto_guide.html
[7] https://airflow.apache.org/docs/apache-airflow/1.10.9/_api/airflow/operators/docker_operator/index.html
[8] https://airflow.apache.org/docs/apache-airflow-providers-http/stable/operators.html
[9] https://airflow.apache.org/docs/apache-airflow/1.10.12/.

Airflow offers a dedicated User Interface (UI), where the execution of the models, the different steps and the system performance can be monitored, as displayed in Figure 4-5. Once the sequence of the models has successfully finished, a relevant notification is sent to the Platform informing it regarding the status.

The Airflow task scheduler enables the user to see the execution status of each model (task) of a scenario (DAG). As shown in Figure 4-6 monitoring the status of all the simulations (i.e., DAG Runs) executed can be observed through the User Interface.

4.4 Connectors of IoT Infrastructure and operational systems

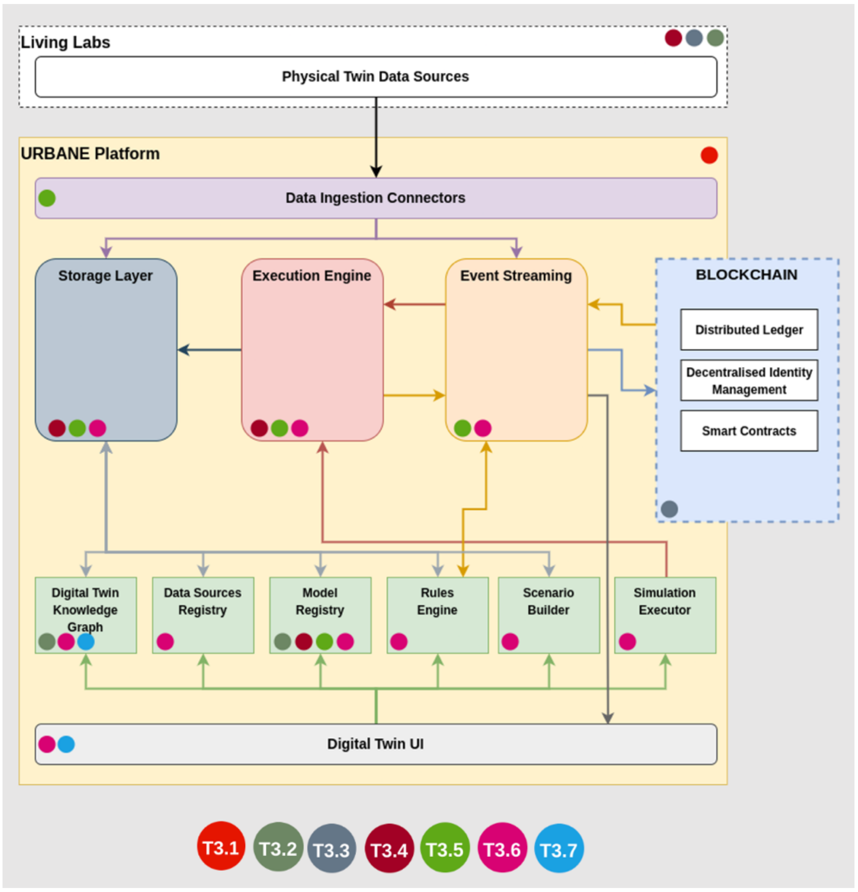


FIGURE 4-7. URBANE DT AND ITS RELATION TO OTHER WP3 BUILDING BLOCKS

Figure 4-7 revisits the URBANE Platform technical architecture that was presented in Figure 4-1. In the latest Figure, however, it illustrates how the different components of the Digital Twin are utilised by the different services. In simple words, it highlights how the different components of WP3 come together to deliver the URBANE Innovation Transferability Platform which is the outcome of the collaborative work undertaken in WP3.

The main building blocks of the URBANE Innovation Transferability Platform are the following:

1. The URBANE Digital Twin Platform (developed by INLE, KNT in the context of Task 3.6).
2. The Blockchain Service (developed by KLU, INLE, KNT in the context of Task 3.3).
3. The URBANE Models (developed by SKEMA, VLTN, NORCE, TU Delft, CERTH under Tasks 3.4, 3.5 and LL Thessaloniki).
4. The Impact Assessment Radar (developed by CERTH under Task 3.7).

As Figure 4-8 illustrates the data ingestion from the LL sources takes place using Connectors, as displayed at the top of the Figure. The Connectors are the entry point into the Big Data Infrastructure of the Platform, getting LL data and events and forwarding them either to dedicated Kafka[10] topics, in case of events, or to the storage layer in the case of batch data. The following paragraph presents the LL Use Cases explored and especially the data pipeline of the Platform, from the moment events are sent by the LL actors until these are displayed to the users through the Platform's User Interface.

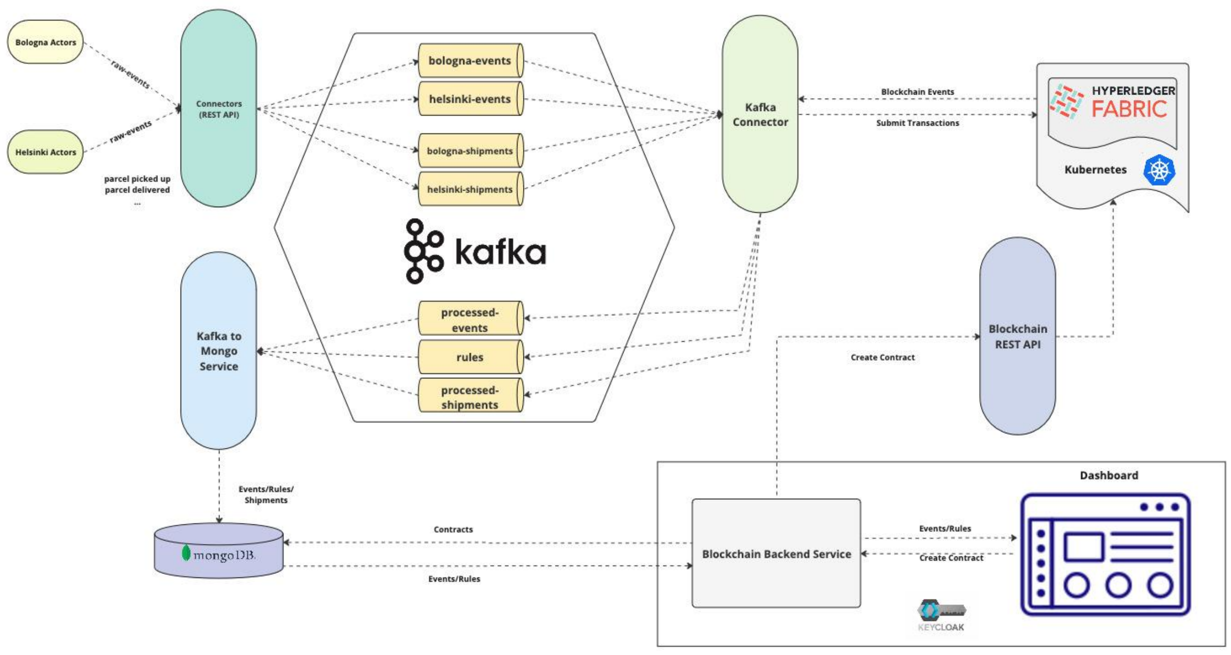


FIGURE 4-8. LL CONNECTORS AND COMMUNICATION WITH BLOCKCHAIN

In URBANE the LL actors send data in the form of real-time *events* to the Platform. These primarily consist of transport instructions/shipments and shipment status events, such as parcel picked up, delivered, or delayed. As illustrated in Figure 4-8, in the context of the Blockchain use cases, the necessary connectors have been developed to enable the ingestion of such events into the Platform.

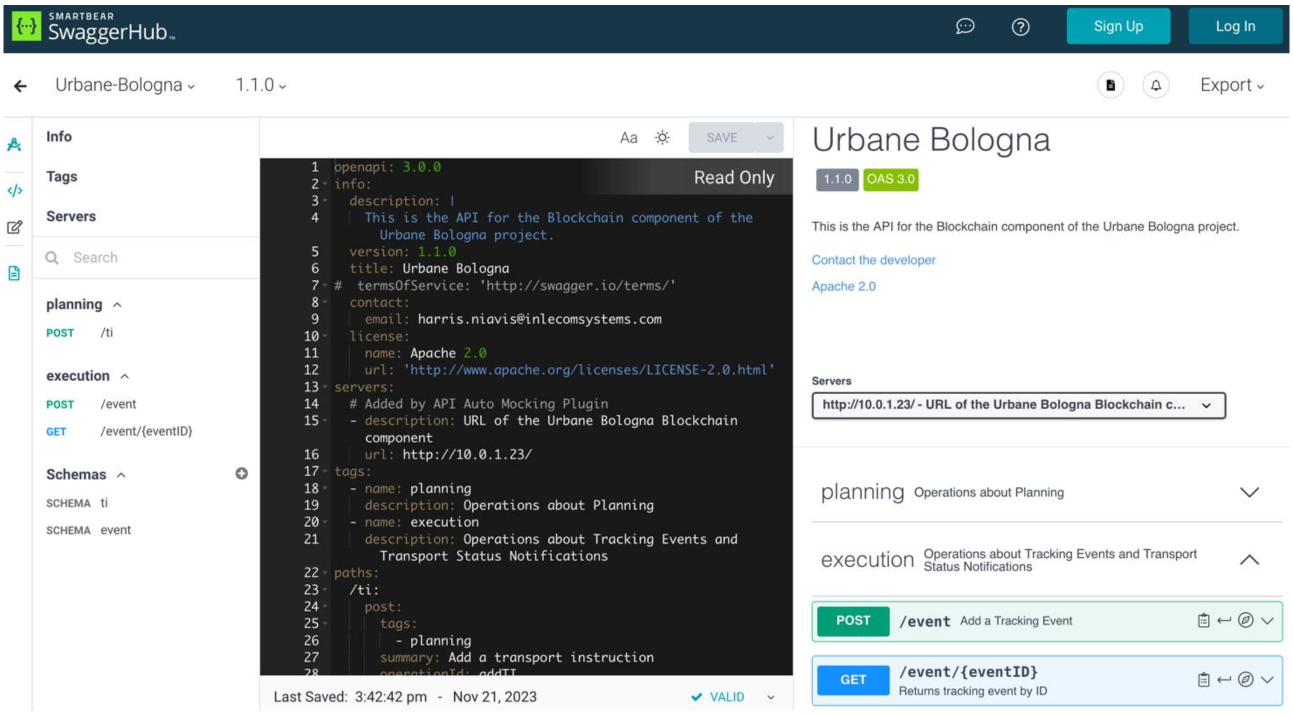
When an event or shipment message is prepared for transfer, a REST request is generated within the connector to facilitate its transmission to a dedicated Kafka topic. An OpenAPI definition was developed using Swagger[11] to design and document the API and in turn ensure the swift consumption of events by the Platform (Figure 4-9). The creation of each Kafka topic is dynamic, depending on the actor initiating the message and the content of the message itself.

The Blockchain Kafka connector component subscribes to the previously mentioned Kafka topics, actively monitoring for incoming messages. Upon receiving a message, it processes it by forwarding it to the Blockchain, where numerous checks are conducted to verify the legitimacy of the actual event or shipment. Some characteristic checks are the following:

- Community Verification: Checks if the channel ID in the contract matches the expected community (i.e., Living Lab).

[10] https://kafka.apache.org/
[11] https://swagger.io/

- Contract Existence: Checks whether the associated contract for this event exists on the ledger.
- Event Existence: Checks whether the event already exists in the contract's list of events.
- Shipment Existence: Checks whether there is an associated shipment for the given event on the ledger.
- Already existing event: Compares all existing events with the newly arrived event to prevent duplication.
- Past or missing events: Checks whether the new event is in the past or whether there are missing events.
- Timestamp Validation: Parses the timestamp for the shipment's start and end dates, and the newly arrived event's timestamp. Checks whether the event's timestamp is between the shipment's start and end dates.



```yaml
openapi: 3.0.0
info:
  description: |
    This is the API for the Blockchain component of the
      Urbane Bologna project.
  version: 1.1.0
  title: Urbane Bologna
#  termsOfService: 'http://swagger.io/terms/'
  contact:
    email: harris.niavis@inlecomsystems.com
  license:
    name: Apache 2.0
    url: 'http://www.apache.org/licenses/LICENSE-2.0.html'
servers:
  # Added by API Auto Mocking Plugin
  - description: URL of the Urbane Bologna Blockchain
    component
    url: http://10.0.1.23/
tags:
  - name: planning
    description: Operations about Planning
  - name: execution
    description: Operations about Tracking Events and
      Transport Status Notifications
paths:
  /ti:
    post:
      tags:
        - planning
      summary: Add a transport instruction
```

Last Saved: 3:42:42 pm - Nov 21, 2023 ✓ VALID

Urbane Bologna

1.1.0 OAS 3.0

This is the API for the Blockchain component of the Urbane Bologna project.

Contact the developer

Apache 2.0

Servers

http://10.0.1.23/ - URL of the Urbane Bologna Blockchain c...

planning Operations about Planning

execution Operations about Tracking Events and Transport Status Notifications

POST /event Add a Tracking Event

GET /event/{eventID} Returns tracking event by ID

FIGURE 4-9. SWAGGER DEFINITION OF URBANE LL EVENTS

Once the Blockchain performs these checks successfully it performs the following actions:

- It stores the event data in the world state under a key derived from the contract and shipment IDs respectively.
- It then emits a processed event with details regarding the contract ID, shipment ID, and event description.
- The Blockchain Kafka Connector then sends the processed event to a designated internal Kafka topic, so that the other components of the Platform can in turn consume it in real time.

Alternatively, should any of the checks or operations fail, the Blockchain returns an error with a corresponding error description.

The Kafka to MongoDB service maintains constant subscription to the internal Kafka topics actively listening for incoming messages. Upon message receival, it stores it within a designated collection in MongoDB, with the storage location determined by the content of the message. Meanwhile, the

Dashboard initiates a polling mechanism, requesting fresh event and shipment data from the backend. The backend searches through the specific MongoDB collections to retrieve the relevant data, which is subsequently transmitted to the Dashboard and presented to the users.

4.5 AI Tools and Big Data Analytics Infrastructure

The URBANE Innovation Transferability Platform offers data processing and storage functionalities as described previously - i.e., section 4 - and displayed in Figure 4-1, through a dedicated data pipeline as shown in Figure 4-2 that allows for the development of our data-driven innovative services to the URBANE LLs.

The Execution Engine of the Platform - shown in Figure 4-1 - allows for the execution of data processing services through dedicated Spark jobs in a Spark cluster. These may range from simple data transformation jobs to advanced AI solutions. In this context, it is worth justifying why Apache Spark[12] is the selected solution for the Big Data Infrastructure of the project. Spark is a multi-language engine that executes data engineering, and Machine Learning (ML) in large scale. Such powerful features, ranging from SQL interactive queries to real-time analytics and ML, constitute Spark as an ideal solution for the URBANE Big Data Infrastructure.

In URBANE a Spark cluster is in charge of running the relevant Spark applications to allow for parallelisation and distribution of the tasks in a set of machines. As Figure 4-10 illustrates, a SparkContext object within the Driver Program (i.e., the main program) coordinates how the different Spark jobs run as independent jobs in the cluster. This is connected to a Cluster Manager (i.e., standalone or Apache Mesos or Hadoop YARN) responsible for resource allocation between applications. Each connected Spark application acquires Executors on the cluster nodes that execute the task that the Driver Program has assigned to them[13].

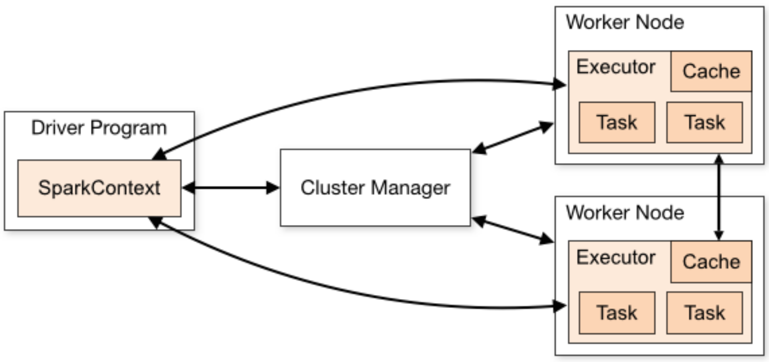


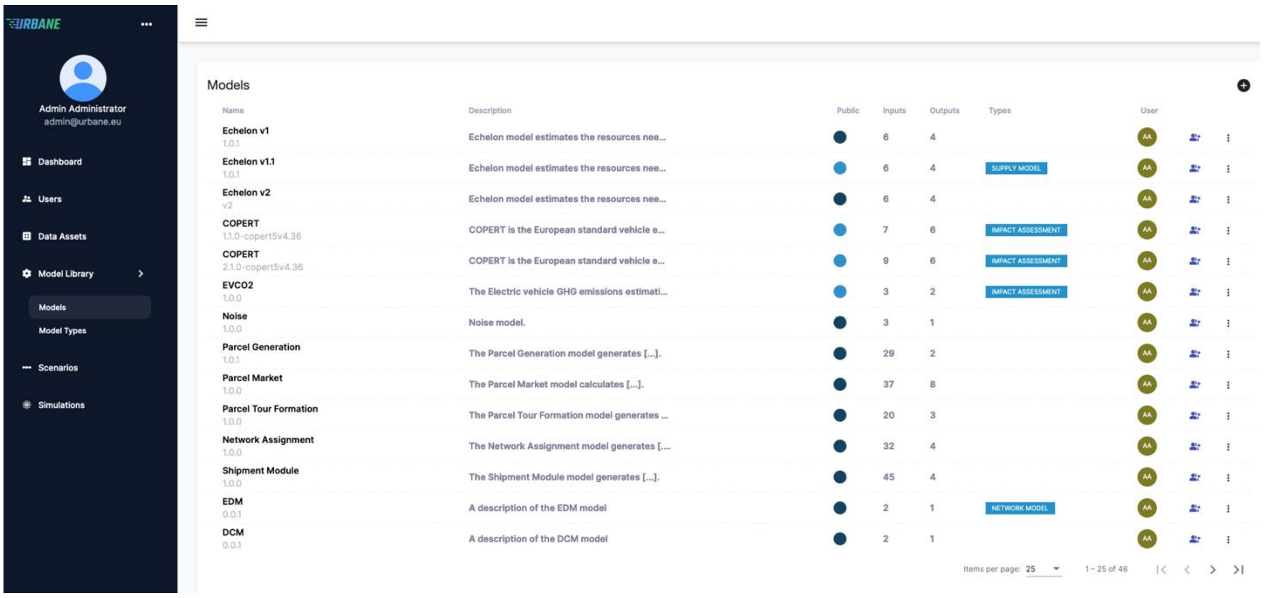
FIGURE 4-10. COMPONENTS OF A SPARK CLUSTER

Apart from offering a Spark cluster for the execution of data processing applications, the Platform allows for the execution of AI models developed by the project partners through its Model Execution Engine. More specifically, AI models are developed in Task 3.5, as presented in section 4.6.1.

All models are integrated in the URBANE Platform based on the model integration requirements (section 4.2.1) through a dedicated integration process (section 4.6) for each model. They can be executed on the

[12] https://spark.apache.org/
[13] https://spark.apache.org/docs/latest/cluster-overview.html

Platform as standalone models or as a sequence of models through the Model Execution Engine that is based on Apache Airflow, as explained previously in section 4.3. Any future model can easily be integrated into the Platform following the model integration process that is explained in 4.2.



Models

Name	Description	Public	Inputs	Outputs	Types	User
Echelon v1 1.0.1	Echelon model estimates the resources nee...	●	6	4		AA
Echelon v1.1 1.0.1	Echelon model estimates the resources nee...	●	6	4	SUPPLY MODEL	AA
Echelon v2 v2	Echelon model estimates the resources nee...	●	6	4		AA
COPERT 1.1.0-copert5v4.36	COPERT is the European standard vehicle e...	●	7	6	IMPACT ASSESSMENT	AA
COPERT 2.1.0-copert5v4.36	COPERT is the European standard vehicle e...	●	9	6	IMPACT ASSESSMENT	AA
EVCO2 1.0.0	The Electric vehicle GHG emissions estimati...	●	3	2	IMPACT ASSESSMENT	AA
Noise 1.0.0	Noise model.	●	3	1		AA
Parcel Generation 1.0.1	The Parcel Generation model generates [...].	●	29	2		AA
Parcel Market 1.0.0	The Parcel Market model calculates [...].	●	37	8		AA
Parcel Tour Formation 1.0.0	The Parcel Tour Formation model generates ...	●	20	3		AA
Network Assignment 1.0.0	The Network Assignment model generates [....	●	32	4		AA
Shipment Module 1.0.0	The Shipment Module model generates [...].	●	45	4		AA
EDM 0.0.1	A description of the EDM model	●	2	1	NETWORK MODEL	AA
DCM 0.0.1	A description of the DCM model	●	2	1		AA

Items per page: 25 1 – 25 of 46

FIGURE 4-11. MODELS AVAILABLE IN URBANE PLATFORM

The Storage layer of the URBANE Platform - displayed in Figure 4-1 - offers storage options for a variety of different data types:

- NoSQL Databases, such as MongoDB[14] for flexible data models whose schema might change in the future. Such is the case of the streaming events sent by the LLs for the Blockchain use cases, as depicted in Figure 4-8.
- Relational Databases, such as PostgreSQL[15], are primarily used for the storage of the models' metadata within the Platform.
- Timeseries solutions, such as Timescale[16] for the processing and storage of timeseries and analytics data.

4.6 Containerisation and integration of models developed in the LLs – Open library of reusable models.

The following sections present the Model Library of URBANE. The library contains models developed in the context of URBANE but also models developed in previous Horizon projects and reused or extended for the URBANE use cases of the Living Labs. Following that, the integration process of each model into the URBANE Platform is presented. Once a model has been successfully integrated, it can be called and executed through the User Interface of the URBANE Platform – as shown in Figure 4-11.

[14] https://www.mongodb.com/
[15] https://www.postgresql.org/
[16] https://www.timescale.com/

4.6.1 URBANE Model Library

The URBANE Model Library is one of the key components of the URBANE Digital Twin platform, as it includes all the digital modelling tools that enable the analysis of 'What-if' scenarios within the Use Cases of the Living Labs. Two categories were identified for the models hoping to provide a good overview of the types of decision support offered by the URBANE Digital Twin based on two criteria:

- model development context and availability, and
- model type according to scientific literature.

Regarding the first category, three distinct types of models can be identified:

- Open-source or open-to-be-reused models,
- Models developed, integrated, and tested in previous projects with high re-use potential,
- URBANE Horizontal models.

The first category refers to digital models in the Transport & Logistics (T&L) sector that are either open-source or, even if closed, they can be reused as standalone executable software programs. One such model is EMISIA's COPERT model[17], whose software implementation is unavailable, but can be re-used as an executable software program, deriving emissions calculations based on a set of input parameters. Other models within the first category are open-source (and, thus, open-to-be-reused) models, just as *jsprit*, an open-source toolkit for solving rich Traveling Salesman Problems (TSP) and Vehicle Routing Problems (VRP). Within the second category of models, which have been developed in previous projects and are expected to have high re-use potential, it can be founded a model such 2-echelon. Based on Daganzo's distance estimation formula[18], the model can provide last-mile logistics services operators with a rough estimation of the resources required to distribute in a specific zone within an urban environment. Finally, a short reference to the third category of models includes the URBANE Horizontal models, developed within URBANE to satisfy the URBANE LLs UCs and demonstrate transferability.

A second classification of models can take place based on the model type. The LEAD project has worked on a classification of models according to the literature that led to the following categories:

- Agent Based Models (ABMs), that simulate individual agent's behaviour.
- Optimisation models, that deal with the minimisation of a cost-related function (e.g., routing models, facility location models, or other Linear Programming, Mixed Integer-Linear Programming models).
- Network models, that handle the allocation and assignment of demand to available network resources.
- Demand models that focus on obtaining and analysing the preferences of agents (including demand synthesis and demand prediction via AI techniques).
- Impact assessment that analyses and measures the impact of different interventions on the last-mile network.

In URBANE, two more categories of models are added to the Model Library:

- Supply models [3] that can be used to predict the level of service of the last-mile network based on network characteristics (e.g., fleet specification) and estimated demand.

[17] COPERT model by EMISIA S.A., the EU standard vehicle emissions calculator, https://www.emisia.com/utilities/copert/

[18] Daganzo's distance estimation formula provides a practical method for approximating average travel distances in vehicle routing and urban logistics models. For details, see: Daganzo, C. F. (1984). *The distance traveled to visit N points with a maximum of C stops per vehicle: An analytic model and an application.* Transportation Science, 18(4), 331–350. Available at: https://doi.org/10.1287/trsc.18.4.331

- Socio-cognitive agent-based models [4] that measure how the adoption of modern policies and interventions, such as crowd shipping, affect the performance of last-mile logistics networks.

With these two classifications in mind, in the next few paragraphs, we proceed with a description and an execution workflow of the main models expected used by the URBANE DT Platform.

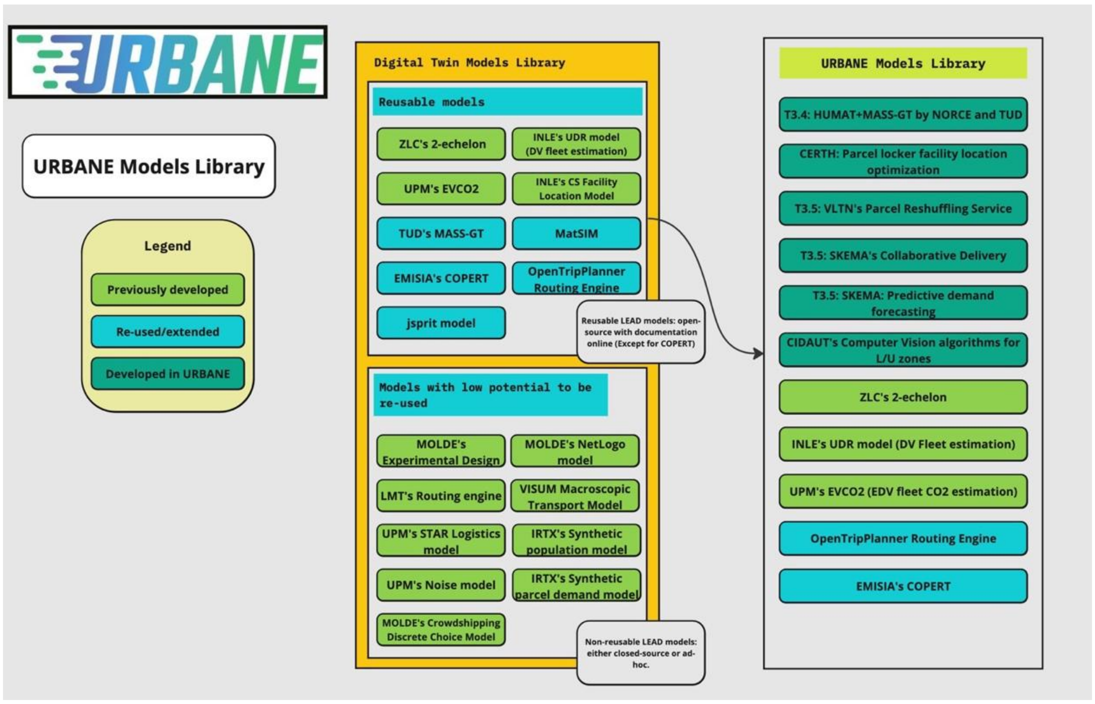
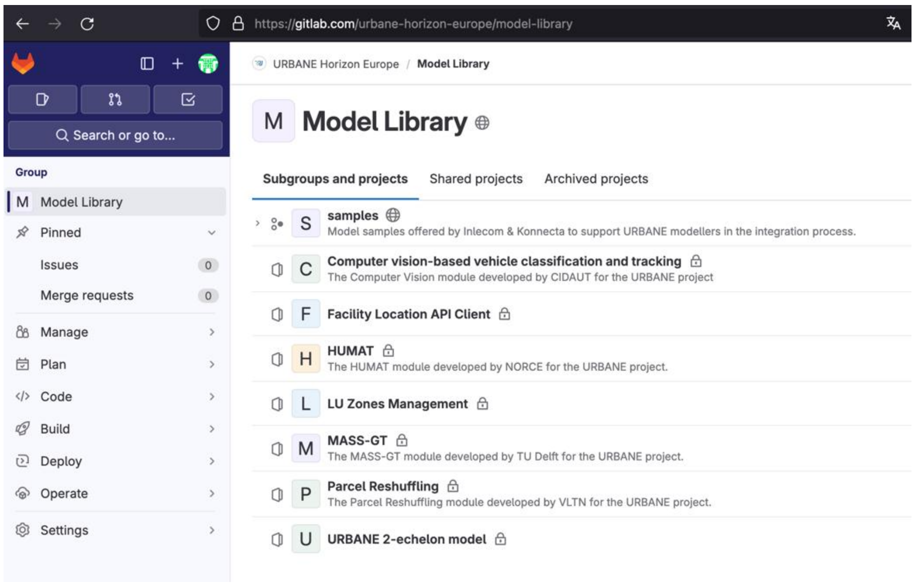

URBANE
URBANE Models Library
Legend
Previously developed
Re-used/extended
Developed in URBANE
Digital Twin Models Library
Reusable models
ZLC's 2-echelon
INLE's UDR model (DV fleet estimation)
UPM's EVCO2
INLE's CS Facility Location Model
TUD's MASS-GT
MatSIM
EMISIA's COPERT
OpenTripPlanner Routing Engine
jsprit model
Reusable LEAD models: open-source with documentation online (Except for COPERT)
Models with low potential to be re-used
MOLDE's Experimental Design
MOLDE's NetLogo model
LMT's Routing engine
VISUM Macroscopic Transport Model
UPM's STAR Logistics model
IRTX's Synthetic population model
UPM's Noise model
IRTX's Synthetic parcel demand model
MOLDE's Crowdshipping Discrete Choice Model
Non-reusable LEAD models: either closed-source or ad-hoc.
URBANE Models Library
T3.4: HUMAT+MASS-GT by NORCE and TUD
CERTH: Parcel locker facility location optimization
T3.5: VLTN's Parcel Reshuffling Service
T3.5: SKEMA's Collaborative Delivery
T3.5: SKEMA: Predictive demand forecasting
CIDAUT's Computer Vision algorithms for L/U zones
ZLC's 2-echelon
INLE's UDR model (DV Fleet estimation)
UPM's EVCO2 (EDV fleet CO2 estimation)
OpenTripPlanner Routing Engine
EMISIA's COPERT

FIGURE 4-12. MODELS INCLUDED IN URBANE MODEL LIBRARY

Figure 4-12 presents the URBANE Model Library that contains models currently in development in the context of URBANE but also reusable models from previous Horizon projects (i.e., the LEAD project[19]).

[19] https://www.leadproject.eu/


https://gitlab.com/urbane-horizon-europe/model-library

URBANE Horizon Europe / Model Library

Model Library

Subgroups and projects | Shared projects | Archived projects

S	samples Model samples offered by Inlecom & Konnecta to support URBANE modellers in the integration process.
C	Computer vision-based vehicle classification and tracking The Computer Vision module developed by CIDAUT for the URBANE project
F	Facility Location API Client
H	HUMAT The HUMAT module developed by NORCE for the URBANE project.
L	LU Zones Management
M	MASS-GT The MASS-GT module developed by TU Delft for the URBANE project.
P	Parcel Reshuffling The Parcel Reshuffling module developed by VLTN for the URBANE project.
U	URBANE 2-echelon model

FIGURE 4-13. THE URBANE MODEL LIBRARY ON GITLAB

4.6.2 Model execution workflows enabled by the DT Platform

Reusability and transferability are key elements of the URBANE Innovation Transferability Platform and the overall URBANE project. Hence, reusing models from previous Horizon projects and applying to the project's LLs is of utmost importance. However, models developed in URBANE also need to have these features, as they will be applied to the Lighthouse LLs, but are also expected to be transferred to the Twinning LLs. The following paragraphs introduce the models available to URBANE stakeholders in the Digital Twin Model Library, explain how they can be used in real business cases, and outline the main integration details.

PI-inspired collaborative routing models

In Task 3.5 two routing models are under development looking to support the transition to the Physical Internet (PI). The models are the "Parcel Reshuffling Service", developed by VLTN, and the "Collaborative deliveries routing" model, developed by SKEMA. Both models conceptually underline one of the main benefits towards the Physical Internet, that is collaboration in delivery between different Logistics Services Providers(LSPs).

The Parcel Reshuffling Service was previously developed in the PLANET project[20]. The model offers collaborative delivery for delivery rounds that could not be successfully completed. Instead of returning the goods to a warehouse, fleet operators must manually identify and assess possible help rounds that share some of the load of the late running round. Following that, they need to identify a location for the exchange, and the drivers typically design the remainder of the route (Figure 4-14).

[20] Horizon 2020 PLANET, Progress towards Federated Logistics Through The Integration Of TEN-T into A Global Trade Network, https://www.planetproject.eu/

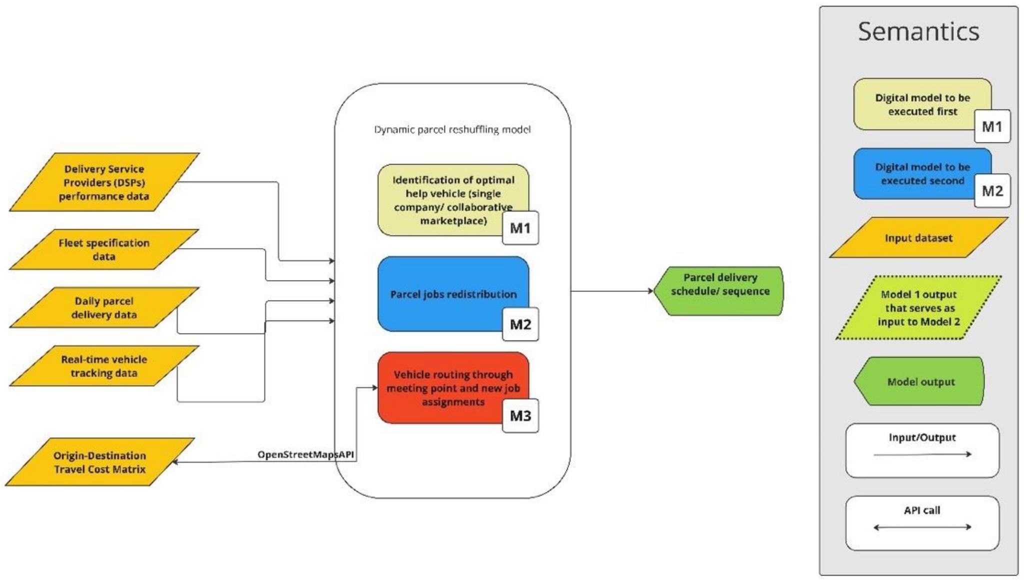

Delivery Service Providers (DSPs) performance data
Fleet specification data
Daily parcel delivery data
Real-time vehicle tracking data
Origin-Destination Travel Cost Matrix
OpenStreetMapsAPI
Dynamic parcel reshuffling model
Identification of optimal help vehicle (single company/ collaborative marketplace)
M1
Parcel jobs redistribution
M2
Vehicle routing through meeting point and new job assignments
M3
Parcel delivery schedule/ sequence
Semantics
Digital model to be executed first
M1
Digital model to be executed second
M2
Input dataset
Model 1 output that serves as input to Model 2
Model output
Input/Output
API call

FIGURE 4-14. MODEL EXECUTION WORKFLOW FOR VLTN'S PARCEL RESHUFFLING SERVICE

The second model within Task 3.5 (developed by SKEMA) focuses on collaborative deliveries routing, enabling the provision of a routing platform, working at two levels: *i)* assignment of parcels to delivery service providers based on a set of incentives, *ii)* development of a solution to a collaborative vehicle routing problem. This model encompasses the first version of a Dynamic service pricing and booking model. Figure 4-15 presents the model execution workflow for this first version of the model. The collaborative deliveries model is expected to be applied to the Bologna LL, and potentially to the Thessaloniki LL.

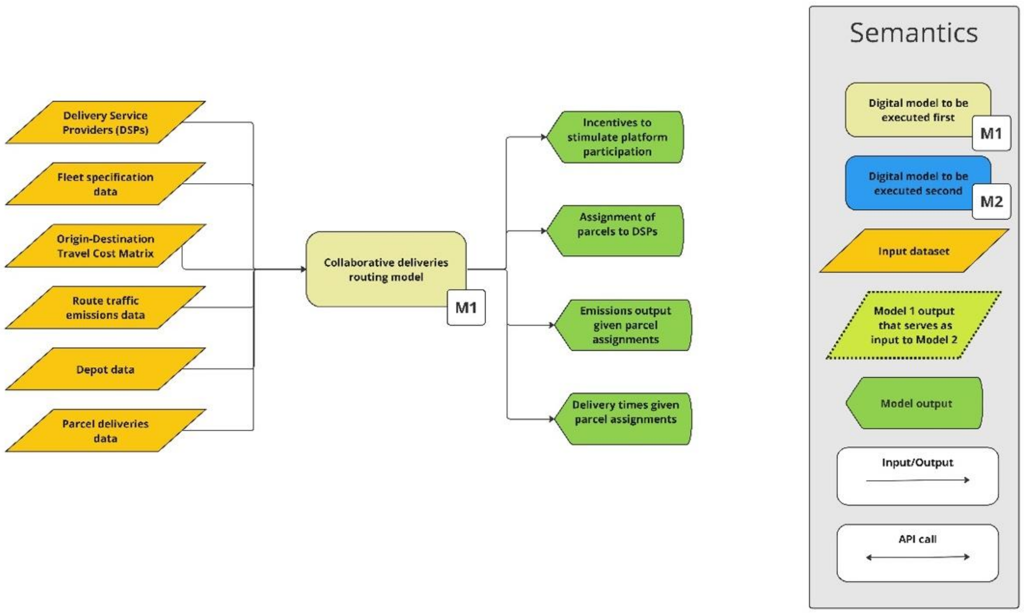

Delivery Service Providers (DSPs)
Fleet specification data
Origin-Destination Travel Cost Matrix
Route traffic emissions data
Depot data
Parcel deliveries data
Collaborative deliveries routing model
M1
Incentives to stimulate platform participation
Assignment of parcels to DSPs
Emissions output given parcel assignments
Delivery times given parcel assignments
Semantics
Digital model to be executed first
M1
Digital model to be executed second
M2
Input dataset
Model 1 output that serves as input to Model 2
Model output
Input/Output
API call

FIGURE 4-15. THE MODEL EXECUTION WORKFLOW FOR SKEMA'S COLLABORATIVE DELIVERIES ROUTING MODEL

Reusable Models

Several models have already been successfully integrated into the Digital Twin in the context of previous Horizon 2020 project (i.e., LEAD project). These models were applied to offer effective decision support in the LEAD project's Living Labs. From this pool of models, some are suitable for re-use within URBANE. Such examples are the 2-echelon[21], copert[22] and EVCO2 models. These models assisted LSPs within the LEAD project to decide on the number of electric and/or conventional delivery vehicles required to cover demand, as well as their expected environmental impact.

Socio-cognitive agent-based simulation with HUMAT and MASS-GT

In Task 3.4, a socio-cognitive agent-based simulation model was developed by combining HUMAT and MASS-GT. This model is designed to test how interventions in last-mile delivery networks depend on public adoption. For example, in crowd-shipping, deliveries can only succeed if non-professional couriers (called bringers) choose to participate. Adoption, however, is not uniform. Different groups of potential bringers, shaped by their local context and social background, may react in very different ways.

Understanding these adoption patterns is therefore essential, thus URBANE's socio-cognitive ABM links HUMAT model. In more detail HUMAT which instantiates a synthetic population with motives and social networks to simulate how consumers evaluate delivery options and influence each other over time, outputting preferred fulfilment choices and satisfaction KPIs—with MASS-GT, which translates parcel demand into logistics operations through its Parcel Demand, Parcel Market and Parcel Scheduling modules.

The framework has already been tested in URBANE Wave 1 Living Labs cities. In Thessaloniki, it was used to analyse parcel locker adoption. HUMAT modelled how consumer choices shifted, while MASS-GT managed locker capacity and redirected excess parcels to traditional couriers. In Helsinki, HUMAT simulated consumer acceptance of autonomous delivery vehicles. MASS-GT, together with a Vehicle Routing Problem (VRP) tool, then measured the operational impacts. Finally, the same modular interface (i.e., Modeling Framework) is designed for transferability in URBANE Wave 2 Living Lab of Karlsruhe. The Wave 2 Living Lab will re-use these calibrated components from the Wave 1 Living Labs, yet successful application will require the integration of local data and additional calibration to reflect the specific social and logistical conditions of the city. Lastly, the Model Execution Workflow is summarised in Figure 4-16.

[21] https://github.com/Horizon-LEAD/2echelon
[22] https://copert.emisia.com/

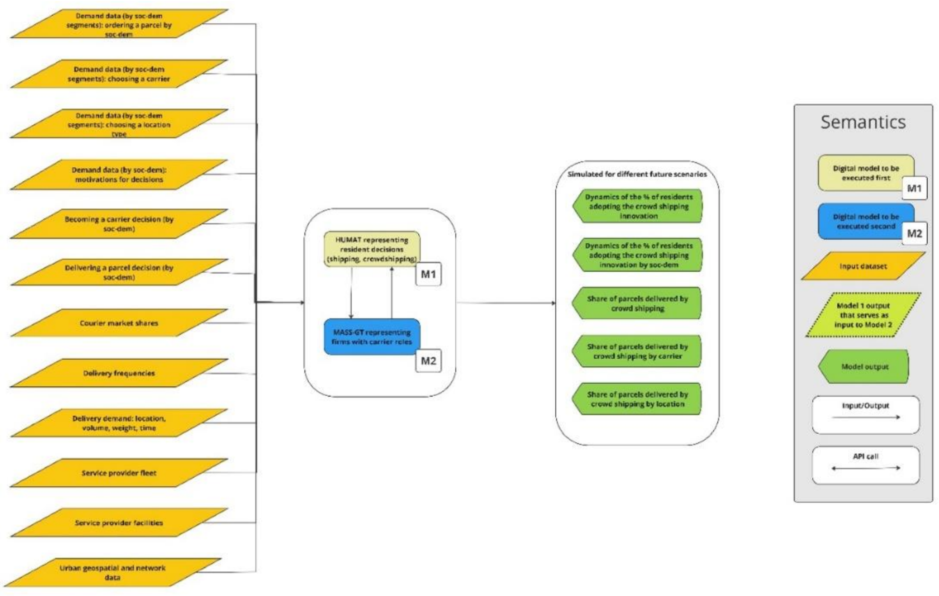

Demand data (by soc-dem segments): ordering a parcel by soc-dem
Demand data (by soc-dem segments): choosing a carrier
Demand data (by soc-dem segments): choosing a location type
Demand data (by soc-dem): motivations for decisions
Becoming a carrier decision (by soc-dem)
Delivering a parcel decision (by soc-dem)
Courier market shares
Delivery frequencies
Delivery demand: location, volume, weight, time
Service provider fleet
Service provider facilities
Urban geospatial and network data
HUMAT representing resident decisions (shipping, crowdshipping)
M1
MASS-GT representing firms with carrier roles
M2
Simulated for different future scenarios
Dynamics of the % of residents adopting the crowd shipping innovation
Dynamics of the % of residents adopting the crowd shipping innovation by soc-dem
Share of parcels delivered by crowd shipping
Share of parcels delivered by crowd shipping by carrier
Share of parcels delivered by crowd shipping by location
Semantics
Digital model to be executed first
M1
Digital model to be executed second
M2
Input dataset
Model 1 output that serves as input to Model 2
Model output
Input/Output
API call

FIGURE 4-16. THE MODEL EXECUTION WORKFLOW FOR THE HUMAT AND MASS-GT MODEL FOR THE SOCIO-COGNITIVE ABM SIMULATION

Parcel Locker and Facility Location Models

In the context of URBANE some LLs develop or utilise their own modelling solutions based on their Use Case requirements and scope. Such a case is the Thessaloniki LL were CERTH used its Shared Parcel Locker and the Facility Location Model, which were applied in Thessaloniki.

The facility location model developed under the URBANE project is a decision-support tool for the optimal spatial allocation of parcel locker networks in urban and peri-urban environments. Based in classical facility location theory and enriched with urban logistics-specific constraints, the model identifies optimal locations for locker installations by balancing service accessibility, operational efficiency, and sustainability goals.

It is particularly suited for designing decentralized last-mile delivery networks by accounting for spatial demand distributions, urban socioeconomic characteristics, and policy-relevant constraints such as walking distance thresholds and land-use limitations. This model supports both single-operator and alliance (shared) configurations, making it applicable to advancing logistics ecosystems that favor interoperability and public-private collaboration.

The model operates using key input parameters such as geocoded demand data (e.g., parcel delivery volumes or population density), candidate facility locations, maximum acceptable walking distances, and installation and operational cost factors. Its outputs include the optimal subset of parcel locker locations, their corresponding service areas, and key performance indicators such as average walking distance, and expected loading factors according to queueing variables accounted in the model. The input parameters and datasets, as well as expected decision support outcomes, are depicted in the Model Execution Workflow in Figure 4-17.

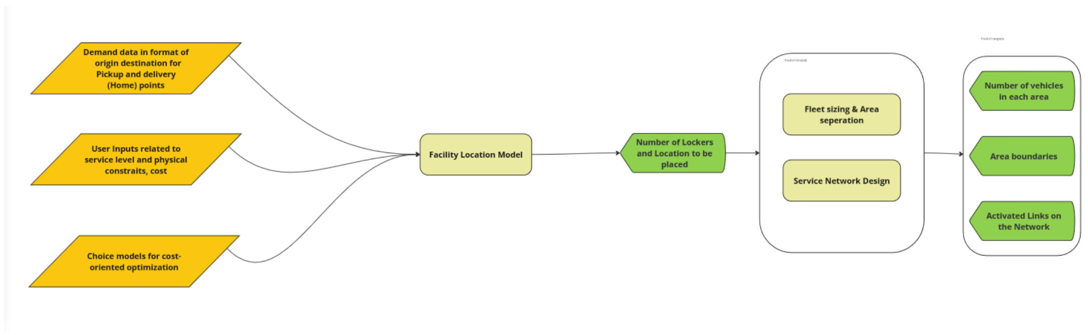

Demand data in format of origin destination for Pickup and delivery (Home) points
User inputs related to service level and physical constraits, cost
Choice models for cost-oriented optimization
Facility Location Model
Number of Lockers and Location to be placed
Fleet sizing & Area seperation
Service Network Design
Number of vehicles in each area
Area boundaries
Activated Links on the Network

FIGURE 4-17. PARCEL LOCKER FACILITY LOCATION OPTIMISATION (CERTH)

__Capacitated Vehicle Routing Problem for E-bikes__

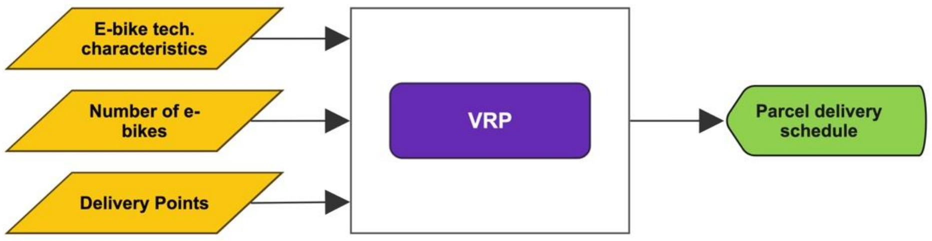
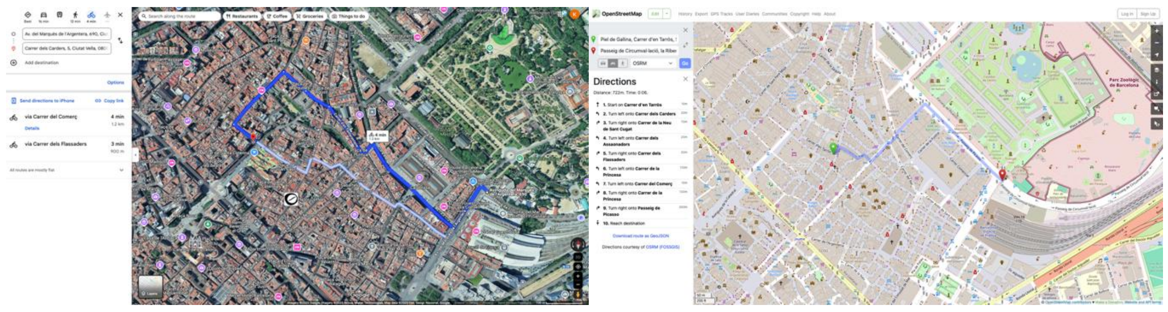


FIGURE 4-18. MODEL EXECUTION WORKFLOW FOR CAPACITATED VRP

In the context of the Barcelona Living Lab, the requirement to develop a Vehicle Routing Model was raised to plan the parcel deliveries in the historical centre (Figure 4-18). The streets are mostly pedestrian zones and not accessible by car. The deliveries are made predominantly by e-cargo bikes, that can use both the roads and pedestrianized streets. It is essential that this feature is captured in the route design to provide an accurate delivery route. To this end, the input for the vehicle routing problem that identifies the optimal stop sequence is an Origin-Destination (OD) matrix, that contains the distances (or times) between any delivery location.

For the design of the OD matrix, various GIS based routing services were considered as well as straight line and Manhattan distances. The cycling routes outputs are often found to differ and therefore further analysis is undertaken. Figure 4-19 illustrates a comparison of the results obtained for a route from the depot to the same delivery location from Google Maps, Open Street Maps and the Open Routing Service.



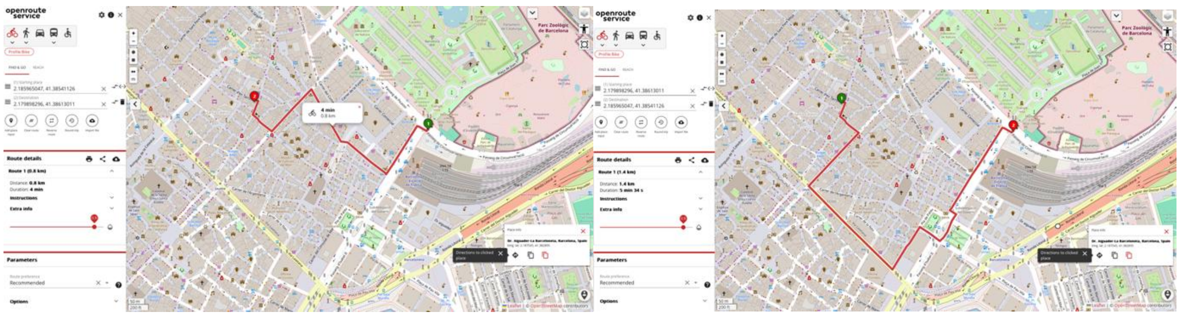


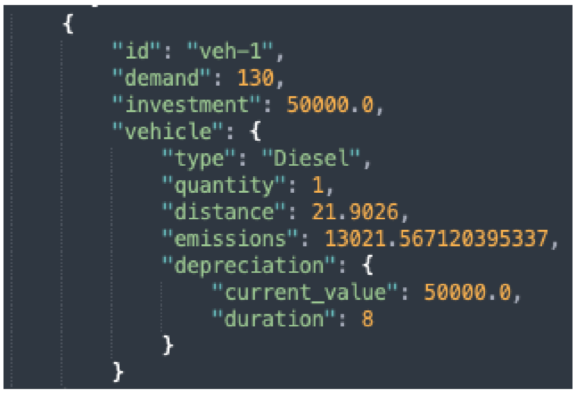
FIGURE 4-19. ROUTE TO DEPOT ROUTES WITH DIFFERENT SERVICES

For the purposes of the demonstrator the Open Routing Service has been identified as the most appropriate tool. Depending on the number of delivery locations, the algorithm can make several requests to the service's API to populate a comprehensive time and distance matrix.

Once the time and distance matrixes are obtained the capacitated VRP is solved considering the e-cargo bikes' carrying capacity. The VRP output is the sequence of delivery locations for each e-cargo bike available in the operator's fleet. Additionally, constraints such as shift duration, and battery capacity (in km) are not considered in the analysis, as they are not hard constraints for the operator. The outcomes of the model are presented in the CitIQore Dashboard, as explained in section 5.3.

Multimodal Cost Benefit Analysis

As in the original version, presented previously in *D3.3 AI-driven models and services*, the tool performs the cost benefit analysis based on two sets of calculations, classified into "daily deliveries cost estimation" and "monthly horizon projections". The daily cost is estimated based on mode-specific utilisation rates that are established from previous analysis and communicated to the Cost Benefit Analysis (CBA) tool through the data structure presented in Figure 4-20. Furthermore, it should be noted that for accommodating the needs and cases of the Wave 2 Living Labs an enhanced version of the CBA tool was developed to better reflect the range of delivery options in the URBANE Living Lab Use Cases. The updated version covers multiple transport modes, vehicle types, and even middle-mile operations. To support this, the tool is designed to be mode-agnostic and expects mode-specific inputs when used. These inputs include transport mode, labour cost per delivery, fuel cost per kilometre, vehicle capacity, and deliveries per staff shift.


```
{
    "id": "veh-1",
    "demand": 130,
    "investment": 50000.0,
    "vehicle": {
        "type": "Diesel",
        "quantity": 1,
        "distance": 21.9026,
        "emissions": 13021.567120395337,
        "depreciation": {
            "current_value": 50000.0,
            "duration": 8
        }
    }
```
FIGURE 4-20. DATA STRUCTURE USED IN CBA

Considering the total number of vehicles in the fleet and the total number of deliveries, the minimum staff s_v and s_d are estimated respectively. The largest of the two values s_v and s_d is then chosen and multiplied by a mode specific daily work rate. The asset depreciation is considered based on the current value and depreciation duration.

The monthly horizon projection takes into account the initial investment cost associated to the operational set-up, a planning horizon and an annual discount rate that enables NPV calculations. The CBA estimates the monthly returns and the investment's break-even point in time. The CBA is employed in all CitIQore interventions, presented in detail in Chapter 5.

4.6.3 Model Integration Process

The integration of each model into the Platform was designed as a process involving several parties. The Business Process Model and Notation (BPMN) diagram depicted in Figure 4-21 presents this process in detail. Three roles can be identified within this process:

(i) the *modeller* who is responsible for the model development,
(ii) the *model integrator* who is responsible for the compliance of the model with the model specifications, and
(iii) the *platform developer* who shall ensure that the model can be and will successfully be integrated into the DT platform.

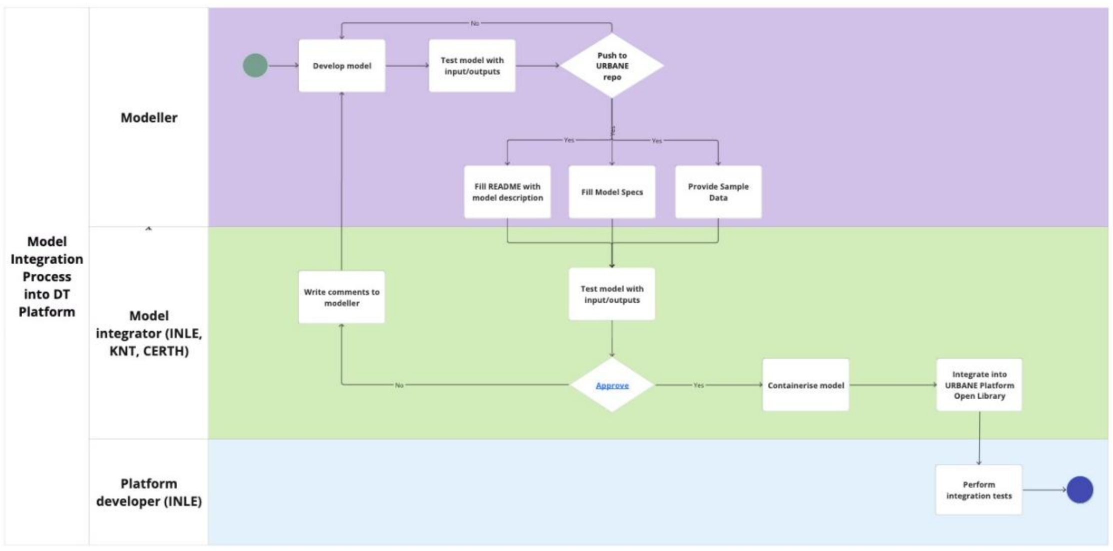

Model Integration Process into DT Platform
Modeller
Model integrator (INLE, KNT, CERTH)
Platform developer (INLE)
No
Develop model
Test model with input/outputs
Push to URBANE repo
Yes
Yes
Fill README with model description
Fill Model Specs
Provide Sample Data
Write comments to modeller
Test model with input/outputs
No
Approve
Yes
Containerise model
Integrate into URBANE Platform Open Library
Perform integration tests

FIGURE 4-21. MODEL INTEGRATION PROCESS INTO URBANE PLATFORM

Figure 4-22 presents a simplified version of this process within the first two columns of the Management Board. The Management Board was designed as a Miro board by the partners involved in T3.6 to support their efforts in the context of the integration process and to ensure that the timeline for all the integrations is followed. To that end, weekly sync meetings were scheduled between the members of the integration team, as well as ad-hoc meetings with each model owner. The Management Board was updated weekly with the latest issues and updates during that meeting.

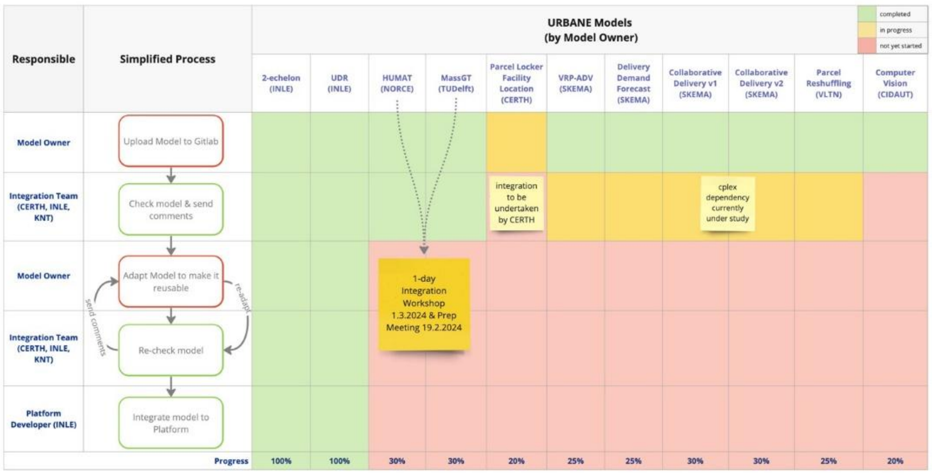

URBANE Models
(by Model Owner)
completed
in progress
not yet started
Responsible
Simplified Process
2-echelon (INLE)
UDR (INLE)
HUMAT (NORCE)
MassGT (TUDelft)
Parcel Locker Facility Location (CERTH)
VRP-ADV (SKEMA)
Delivery Demand Forecast (SKEMA)
Collaborative Delivery v1 (SKEMA)
Collaborative Delivery v2 (SKEMA)
Parcel Reshuffling (VLTN)
Computer Vision (CIDAUT)
Model Owner
Upload Model to Gitlab
Integration Team (CERTH, INLE, KNT)
Check model & send comments
integration to be undertaken by CERTH
cplex dependency currently under study
Model Owner
Adapt Model to make it reusable
1-day Integration Workshop 1.3.2024 & Prep Meeting 19.2.2024
send comments
re-adapt
Integration Team (CERTH, INLE, KNT)
Re-check model
Platform Developer (INLE)
Integrate model to Platform
Progress
100%
100%
30%
30%
20%
25%
25%
30%
30%
25%
20%

Figure 4-22. Model Integration Process & Management Board

The model integration specifications – presented in section 4.2.1 - were communicated to the model owners by the integration team in dedicated meetings during the first months of the project. This also included handing out guidelines for a successful model integration to the model owners in the form of a dedicated document. The guidelines document can be found in Annex I.

Overall, the model owners and the members of the integration team have been in close collaboration so that the models developed meet the expected requirements from a technical and infrastructural standpoint, to be successfully integrated in the platform's Execution Engine. The final step of the integration process involved the containerisation of the models so that they become available on the URBANE Platform. This process was explained in detail in section 4.2.2.

The model integration process consisted of two phases:

- Phase I: December '23 – January '24. During this phase the modelling work was still in progress, hence draft versions of the models were integrated into the DT. The main goal of this phase was to test the integration as a process and to ensure a smooth final integration in the summer of 2024.
- Phase II: Final Integration (summer '24). Once the models were finalised, their final versions were integrated into the DT platform, making them available for execution through the DT portal.

All integrated models are available for execution through the DT Portal (https://urbane.inlecom.eu). Figure 4-23 presents the Dashboard page, present to the user once they sign in, while Figure 4-24 presents the Models page, listing all models integrated and available for execution through the Portal.

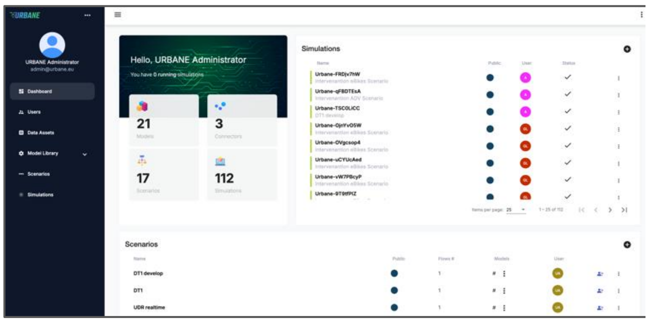


Figure 4-23. URBANE DT Portal Dashboard

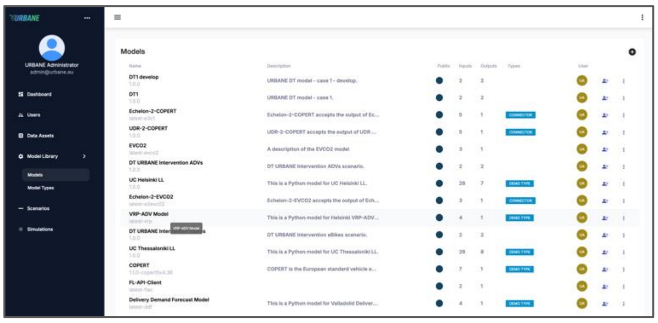


Figure 4-24. URBANE DT Portal. Models Page

5 URBANE CitIQore Application

The work presented in this section addresses the requirements listed in Section 2.4, i.e., requirements raised by the URBANE partners during the first phase of the project. In response to these requirements, a domain-specific application was designed and developed to support stakeholders in interacting with the URBANE Digital Twin Platform more intuitively. This application, named CitIQore, is characterised by ease of use and rich visualisation capabilities, making it accessible even for non-expert users who need to simulate and analyse last-mile logistics scenarios.

The developed application serves as a layer on top of the existing URBANE Digital Twin Platform offering simulation capabilities, utilising (a) models from the Model Library, and (b) historical and real-time data from the project's Living Labs. As illustrated in Figure 5-9, Figure 5-15, Figure 5-20 users interact with the application by selecting different parameters, which are in essence inputs to the models integrated in the Digital Twin. Following that, the application interacts with the Digital Twin by calling a sequence of models using the user-generated input. Within the Digital Twin, the specified sequence of models is executed, and the resulting outputs are returned to the application, where they are made available to users through a dedicated dashboard. The application currently offers three interventions in last-mile logistics that can be applied to any city context across Europe, as shown in Figure 5-1 and they explained in depth in the subsequent sections.

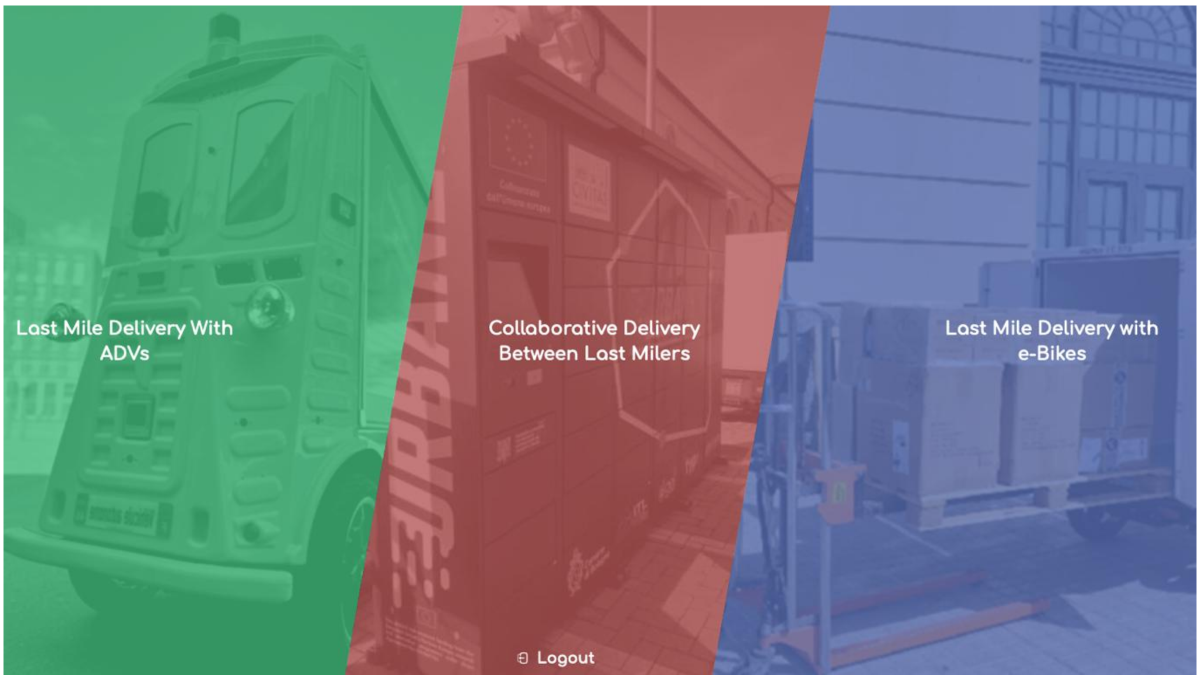

Last Mile Delivery With ADVs
Collaborative Delivery Between Last Milers
Last Mile Delivery with e-Bikes
Logout

Figure 5-1. CitIQore Interventions

Once a user logs in to the CitIQore application, they are called to select between those three available interventions. As transferability is a key requirement in URBANE, CitIQore was designed with that in mind. In this regard, the user may select any area on the map of Europe to place their scenario in (Figure 5-2).

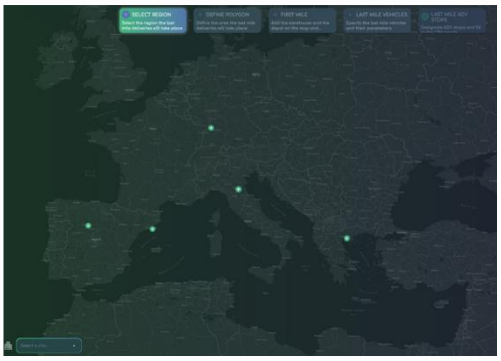


FIGURE 5-2. CHOOSE AREA/CITY ON THE MAP

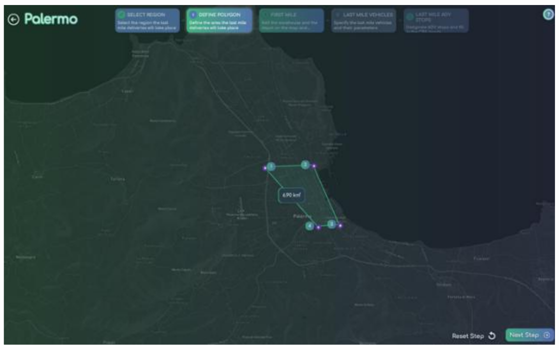


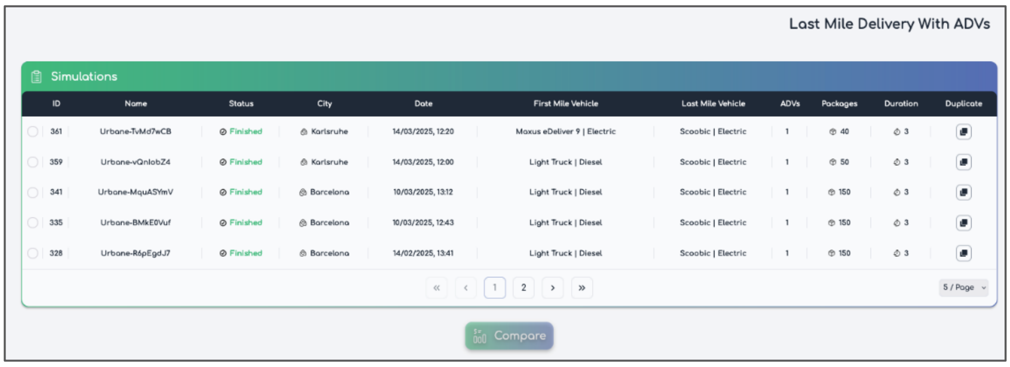
FIGURE 5-3. CHOOSE POLYGO

In each intervention the user is called to define a "what-if" scenario by taking the following steps:

1. Choose an area/city where the scenario will take place (Figure 5-2).
    a. In some interventions they may also specify a polygon of interest (Figure 5-3).
2. If the intervention involves the first mile in the deliveries, select the appropriate first-mile options.
3. Allow the user to select the last mile options.
4. Press the "Calculate" button to allow for the sequence of models to be executed in the DT.

This process is explained in more detail in the following sections, highlighting the main selection criteria for the user per intervention.

Once a simulation has finished successfully it becomes available at the Simulations page (Figure 5-4), where a user may select to view the Dashboard page of a single simulation – as shown in Figure 5-5 - or may select multiple simulations to view a comparative Dashboard to check the differences between different cases (e.g., a simulation focused on electric vehicles compared to one focused on conventional vehicles), as displayed in Figure 5-6.



Last Mile Delivery With ADVs

Simulations

ID	Name	Status	City	Date	First Mile Vehicle	Last Mile Vehicle	ADVs	Packages	Duration	Duplicate
361	Urbane-TvMd7wCB	Finished	Karlsruhe	14/03/2025, 12:20	Maxus eDeliver 9 | Electric	Scoobic | Electric	1	40	3	
359	Urbane-vQnlobZ4	Finished	Karlsruhe	14/03/2025, 12:00	Light Truck | Diesel	Scoobic | Electric	1	50	3	
341	Urbane-MquASYmV	Finished	Barcelona	10/03/2025, 13:12	Light Truck | Diesel	Scoobic | Electric	1	150	3	
335	Urbane-BMkE0Vuf	Finished	Barcelona	10/03/2025, 12:43	Light Truck | Diesel	Scoobic | Electric	1	150	3	
328	Urbane-R6pEgdJ7	Finished	Barcelona	14/02/2025, 13:41	Light Truck | Diesel	Scoobic | Electric	1	150	3	

FIGURE 5-4. LIST OF SIMULATIONS

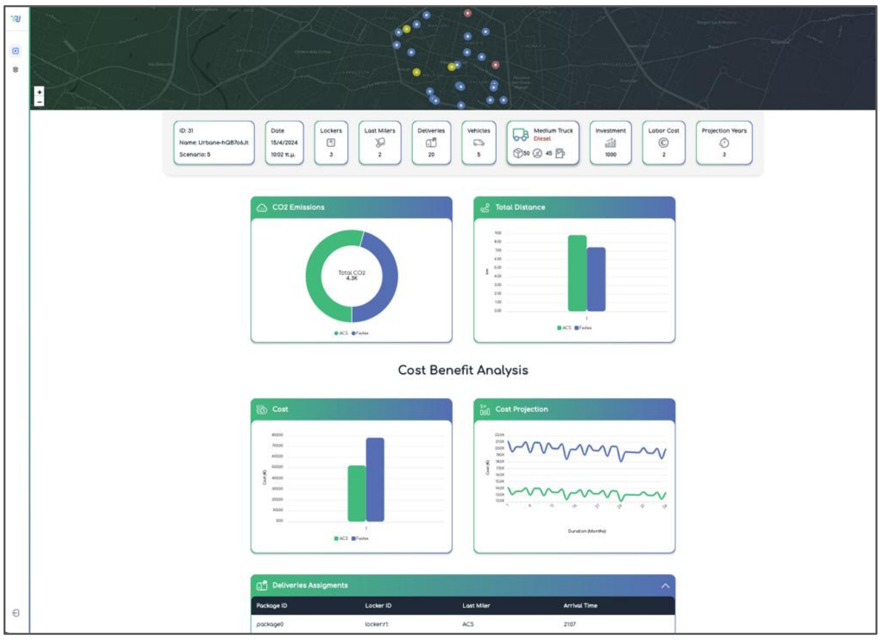


FIGURE 5-5. SINGLE SCENARIO DASHBOARD

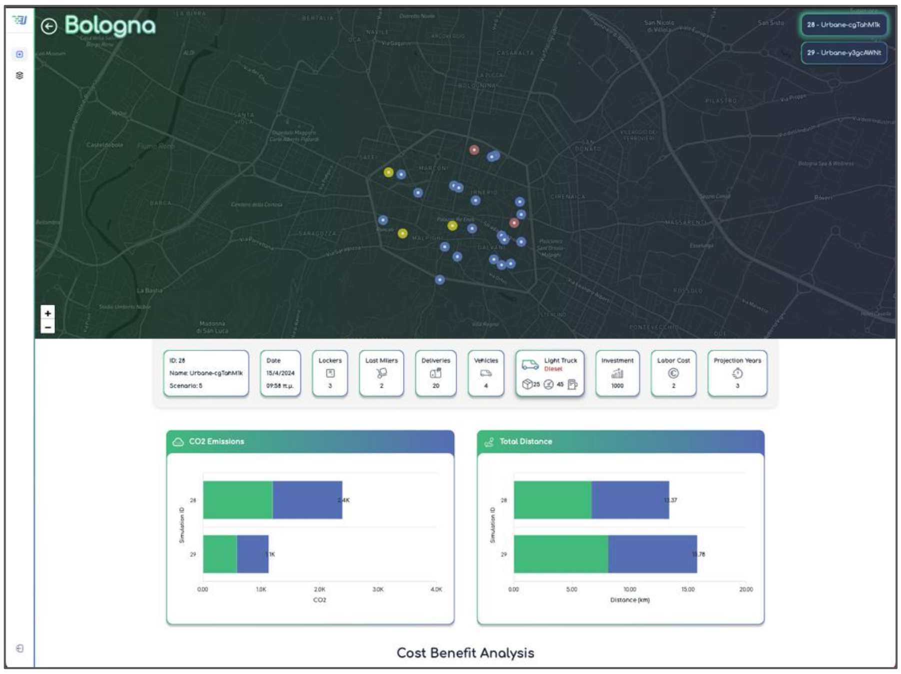


FIGURE 5-6. COMPARATIVE DASHBOARD BETWEEN 2 SCENARIOS

Finally, a user may wish to execute a similar "what-if" scenario by using an existing one and making small modifications to the inputs. This is possible by visiting the corresponding scenario in the List (Figure 5-4) and selecting the "Duplicate" button. Then, they are directed to a prefilled scenario page on the map, where they are called to make modifications and press the "Calculate" button to execute the models with the updated inputs (Figure 5-7). Once the scenario execution has finished and the newly executed scenario is available in the Simulations List page (Figure 5-4), the user can select multiple scenarios and then view a Comparative Dashboard (Figure 5-6) by pressing the "Compare" button.

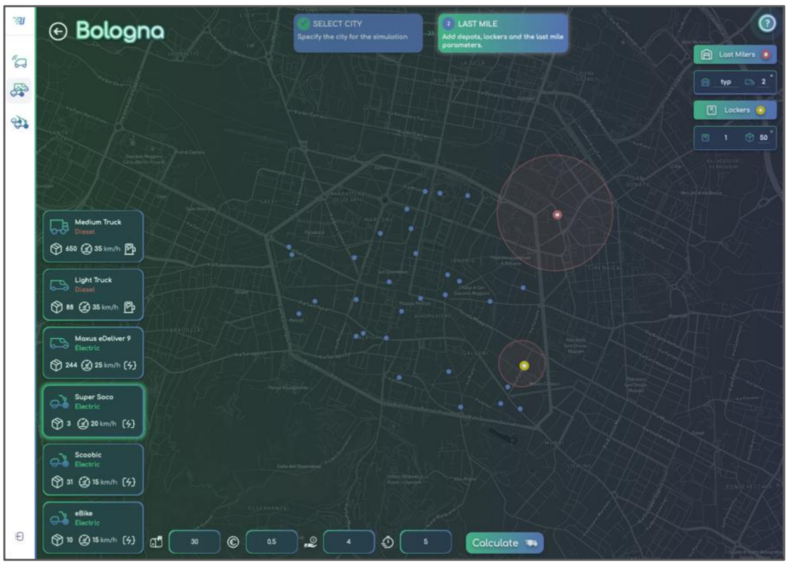


FIGURE 5-7. DUPLICATE FUNCTIONALITY

Finally, CitIQore is available to URBANE consortium partners who have acquired credentials using this link: https://urbane.inlecom.eu/citiqore, while the following sections present each of the three interventions in more detail.

5.1 Collaborative Delivery between Last Milers

The first intervention fosters the introduction of the Physical Internet concept in last mile logistics by examining the collaboration between last-mile logistics operators using shared lockers and the employment of electric vehicles for efficient and emission-free deliveries. Users can test different what-if scenarios on selected cities by choosing different last-mile options (vehicle types, depot and locker placement) and receive insights on CO2 emissions reduction, kilometres driven followed by a cost benefit analysis on their chosen scenario. It was initially developed for the Bologna Living Lab; however, it can be applied to different city contexts through the application's interface.

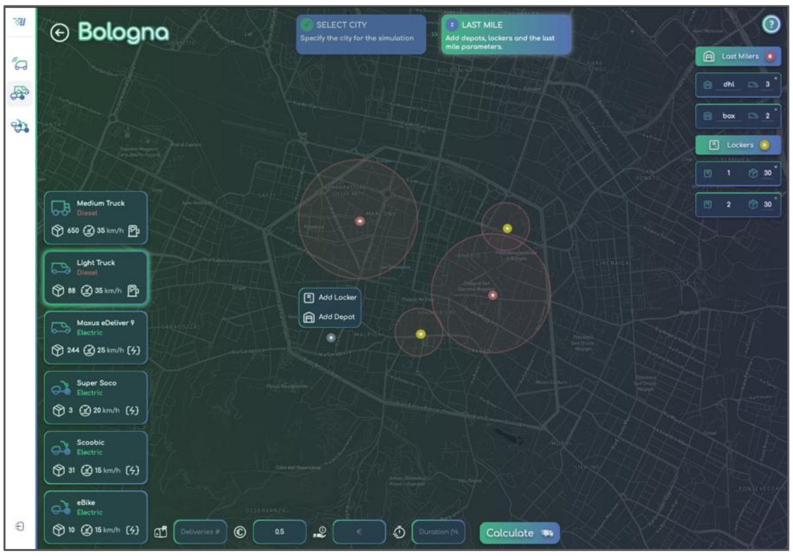


FIGURE 5-8. USER INPUTS PAGE (COLLABORATIVE DELIVERY)

After selecting the city, the user is called to choose the criteria with which the sequence of models should be executed. All inputs are selected directly on a map user interface, as shown in Figure 5-8.

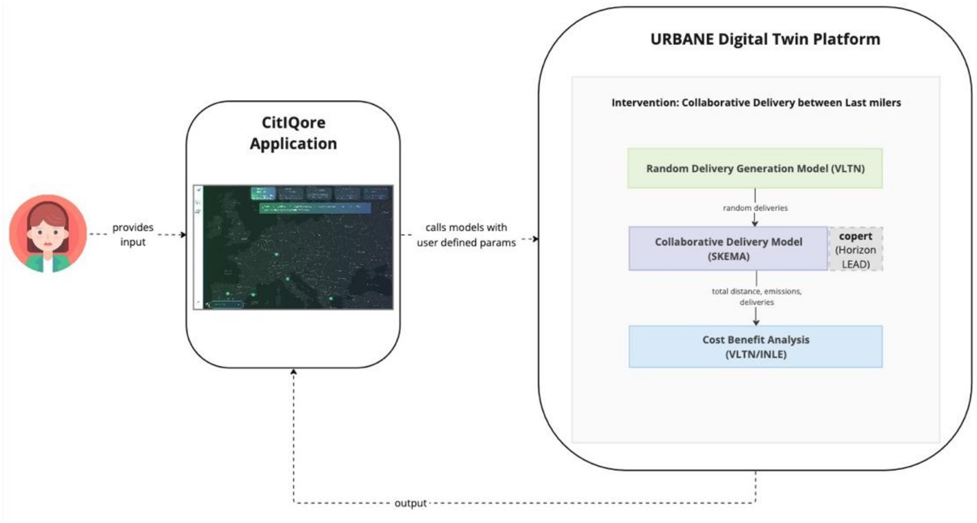

provides input
CitIQore Application
calls models with user defined params
URBANE Digital Twin Platform
Intervention: Collaborative Delivery between Last milers
Random Delivery Generation Model (VLTN)
random deliveries
Collaborative Delivery Model (SKEMA)
copert (Horizon LEAD)
total distance, emissions, deliveries
Cost Benefit Analysis (VLTN/INLE)
output

FIGURE 5-9. CITIQORE INTERACTION DIAGRAM IN COLLABORATIVE DELIVERY

In essence, at this stage the user wishes to ask certain "what-if" scenarios that shall allow them to understand better the potential consequences of certain decisions with regards to last mile logistics: scenarios such as "what if we use electric vehicles in last mile logistics?", "what if we place a depot at a location inside the city centre?" and so on. Having such questions in mind the user may interact with the application by selecting the relevant options (e.g., electric vehicle or Diesel vehicle) and triggering the calculation by the Digital Twin. As shown in Figure 5-9, the following models are executed in sequence in the DT:

- Random Delivery Generation Model, to place random delivery points on the map interface.
- Collaborative Delivery Model to orchestrate the assignment of the parcels to different last milers and lockers (described briefly in section 4.6.2, presented in detail in D3.3 *AI-driven models and services*).

- Copert model[23], to calculate emissions, used previously in Horizon LEAD project.
- Cost Benefit Analysis, to estimate the economic effects of a potential investment decision (described in section 4.6.2, also presented in D3.3 *AI-driven models and services*). This model is used in all three interventions.

Once the simulation has finished, the outputs are presented, as in all interventions, in the Dashboard page (Figure 5-10). A demo video is available here: https://www.youtube.com/watch?v=DKzW82uBVoc

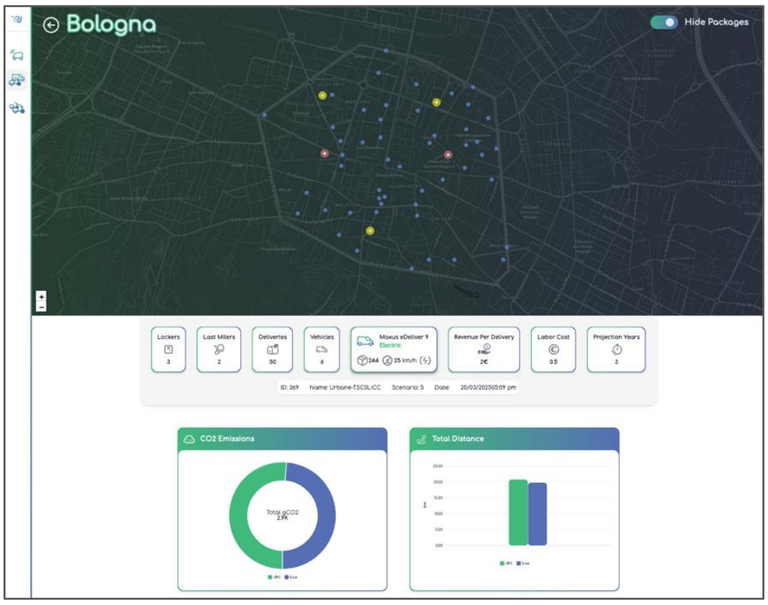


FIGURE 5-10. DASHBOARD (COLLABORATIVE DELIVERY)

5.2 Last-mile Delivery with ADVs

The second intervention demonstrates the integration of Autonomous Delivery Vehicles (ADVs) and electric bikes for last-mile logistics, enabling logistics service providers and cities to evaluate the feasibility and financial sustainability of transitioning to low-emission deliveries. Users can test different 'what-if' scenarios on any location on the map by selecting different first-mile and last-mile options (vehicle types, depot placement etc.). Insights are then provided via the Dashboard page on CO2 emissions reduction, kilometres driven, followed by a cost benefit analysis of their chosen scenario.

This scenario, inspired by the Helsinki Living Lab, consists of several steps, where the user is called to select different input options to formulate their desired scenario to be executed by the DT. The steps are the following, and can be viewed in the following Figures (5-11 to 5-14):

- Step 1. The user is called to select any area/city on the map of Europe and define a specific polygon where they will base their scenario on.
- Step 2. This step focuses on the first mile delivery, expecting the placement of a warehouse outside the polygon area and a depot inside the polygon. Different option for the first mile vehicle can be selected based on their technical characteristics. At this step, the number of deliveries is also inserted.

[23] https://copert.emisia.com/

- Step 3. The last mile delivery takes place using ADVs, however for any parcels outside the ADVs catchment area a secondary support vehicle (electric bike) can be chosen.
- Step 4. In the final step the users are called to add ADV stops across the route(s) of the ADV(s) but also select the inputs for the cost benefit analysis.

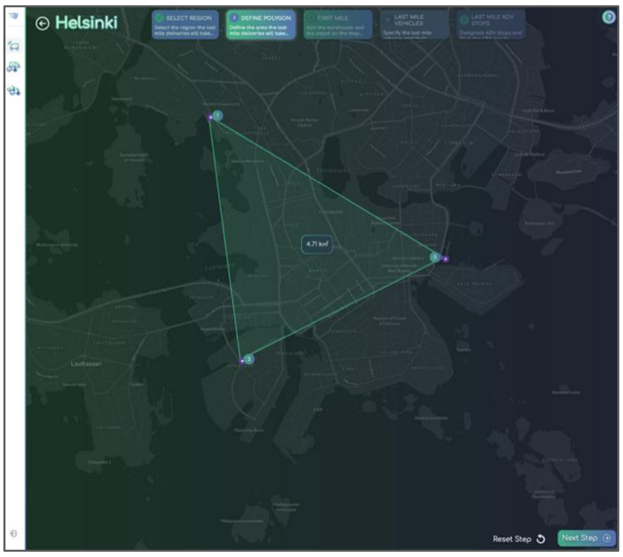


FIGURE 5-11. STEP 1. POLYGON SELECTION

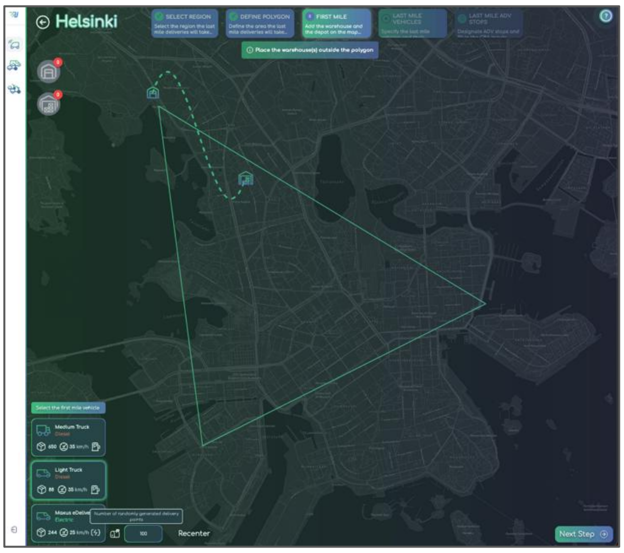


FIGURE 5-12.STEP2. FIRST-MILE

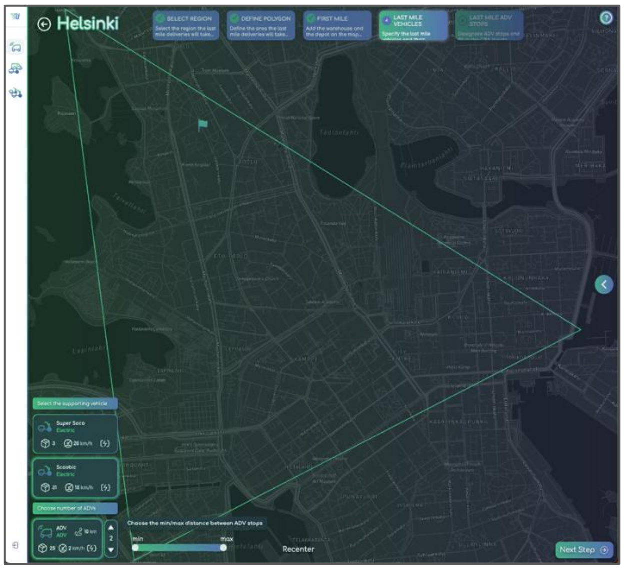


FIGURE 5-13. STEP 3. LAST-MILE VEHICLES

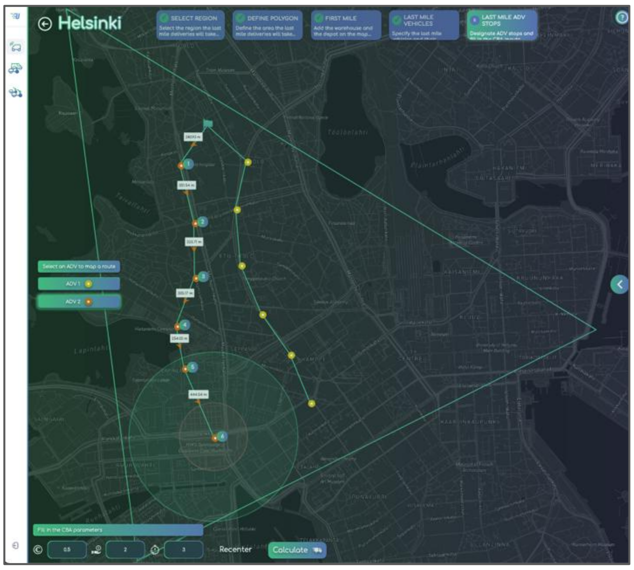


FIGURE 5-14. STEP 4. ADV ROUTE & CBA

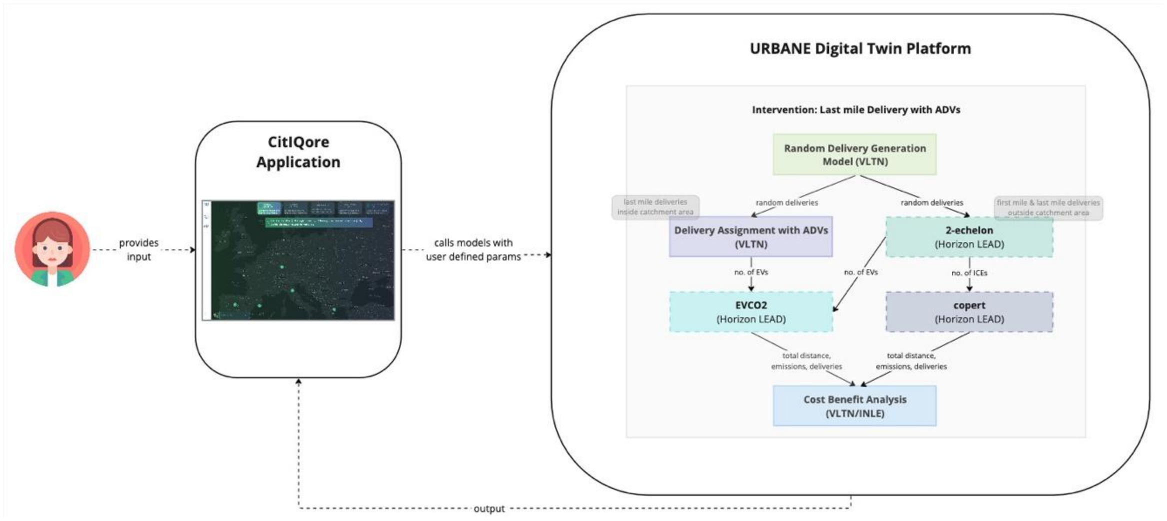
Similarly to the previous intervention, a number of different models are executed in the DT at the moment when the "Calculate" button is pressed in the map interface. As shown in Figure 5-15, these are the following:

- Random Delivery Generation, to generate random delivery points on the map.
- *2-echelon* model (previously employed in Horizon LEAD model) to calculate the number of vehicles in the first mile and last mile (excluding the ADVs) and the total distances they cover.
- Delivery Assignment with ADVs model – described in D3.3 *AI-driven models and services* – that calculates the time at which the robot arrives at each delivery point, the percentage of parcels

successfully delivered, the average waiting time, and the average queue length at each delivery point.

- CO2 emissions calculation. In the case of electric vehicles, the *EVCO2* model is used (model from Horizon LEAD), while in the case of conventional vehicles *copert* is employed.
- Finally, the CBA model estimates the overall financial performance of a potential investment.

A demo video can be found here: https://www.youtube.com/watch?v=k328j2W0jXE

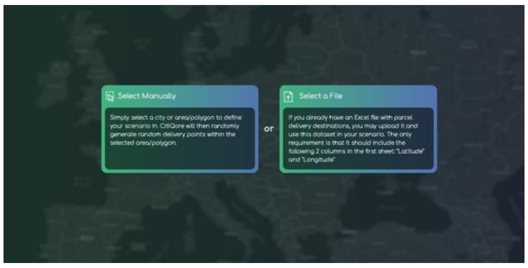


provides input
CitIQore Application
calls models with user defined params
output
URBANE Digital Twin Platform
Intervention: Last mile Delivery with ADVs
Random Delivery Generation Model (VLTN)
last mile deliveries inside catchment area
random deliveries
random deliveries
first mile & last mile deliveries outside catchment area
Delivery Assignment with ADVs (VLTN)
2-echelon (Horizon LEAD)
no. of EVs
no. of EVs
no. of ICEs
EVCO2 (Horizon LEAD)
copert (Horizon LEAD)
total distance, emissions, deliveries
total distance, emissions, deliveries
Cost Benefit Analysis (VLTN/INLE)

FIGURE 5-15.CITIQORE INTERACTION DIAGRAM IN LAST-MILE DELIVERY WITH ADVS

5.3 Last Mile Delivery with E-bikes

*This intervention f*osters the introduction of the Physical Internet concept in last mile logistics through the employment of e-bikes in the last mile for efficient and emission-free deliveries. Users may test different "what-if "scenarios by choosing different first-mile (warehouse and depot placements) and last-mile options (vehicle types) and receive insights on CO2 emissions reduction, *kilometres* driven followed by a *cost benefit* analysis on their chosen scenario. This intervention was inspired by the Barcelona Living Lab.



Simply select a city or area/polygon to define your scenario in. CitiQore will then randomly generate random delivery points within the selected area/polygon.

If you already have an Excel file with parcel delivery destinations, you may upload it and use this dataset in your scenario. The only requirement is that it should include the following 2 columns in the first sheet: "Latitude" and "Longitude"

FIGURE 5-16. RANDOM DELIVERIES OR DELIVERIES PLAN

Apart from offering randomly generated delivery points inside a specific polygon in any location on the map interface, in this intervention users can also upload their own delivery plans to test different "what-if" scenarios with (Figure 5-16).

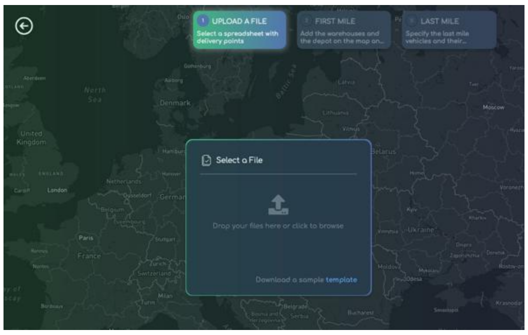

UPLOAD A FILE
Select a spreadsheet with delivery points
FIRST MILE
LAST MILE
Select a File
Drop your files here or click to browse
Download a sample template

FIGURE 5-17. EXCEL IMPORT

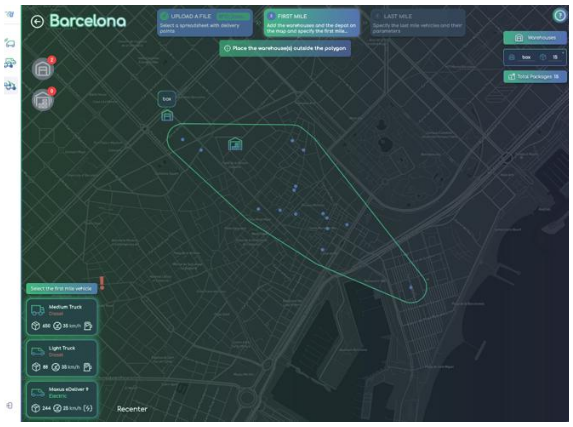


FIGURE 5-18. FIRST MILE OPTIONS

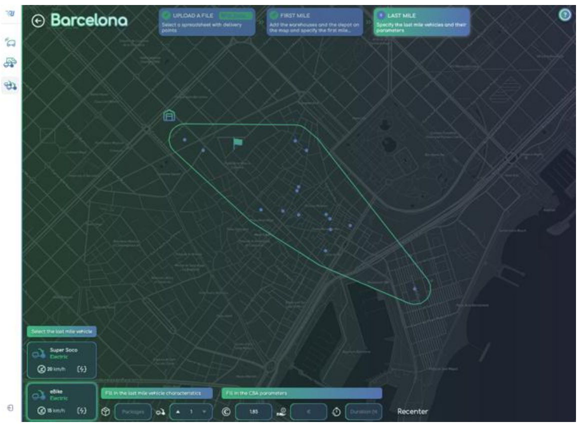

Barcelona
UPLOAD A FILE
FIRST MILE
LAST MILE
Select the last mile vehicle
Super Soco
eBike
Fill in the last mile vehicle characteristics
Fill in the CBA parameters
Recenter

FIGURE 5-19. LAST MILE OPTIONS

As Figure 5-17 indicates the users are expected to upload and Excel spreadsheet, including details such as Latitude and Longitude per delivery. Following that, the intervention consists of two main steps:

- First Mile (Figure 5-18): This step involves the placement of one to three warehouses outside the focus area (representing different first mile stakeholders), the selection of the first-mile vehicle, and the placement of the depot where the vehicles will deliver the parcels to.
- Last Mile (Figure 5-19): At this point, the users select (a) criteria on the last mile vehicles (number of e-bikes, capacity) and (b) the CBA inputs.

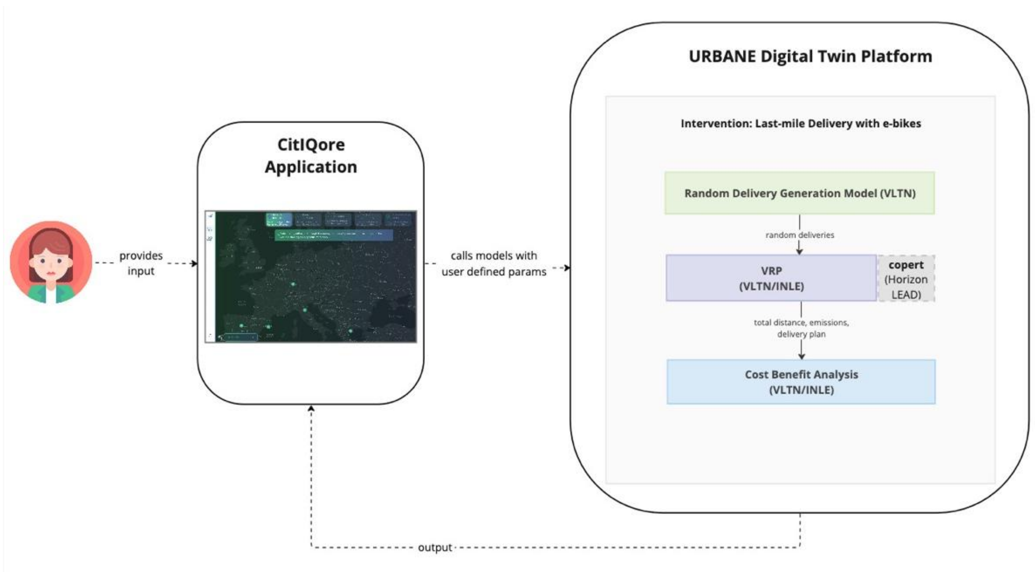

URBANE Digital Twin Platform
Intervention: Last-mile Delivery with e-bikes
Random Delivery Generation Model (VLTN)
random deliveries
VRP (VLTN/INLE)
copert (Horizon LEAD)
total distance, emissions, delivery plan
Cost Benefit Analysis (VLTN/INLE)
CitIQore Application
provides input
calls models with user defined params
output

FIGURE 5-20.CITIQORE INTERACTION DIAGRAM IN LAST-MILE DELIVERY WITH E-BIKES

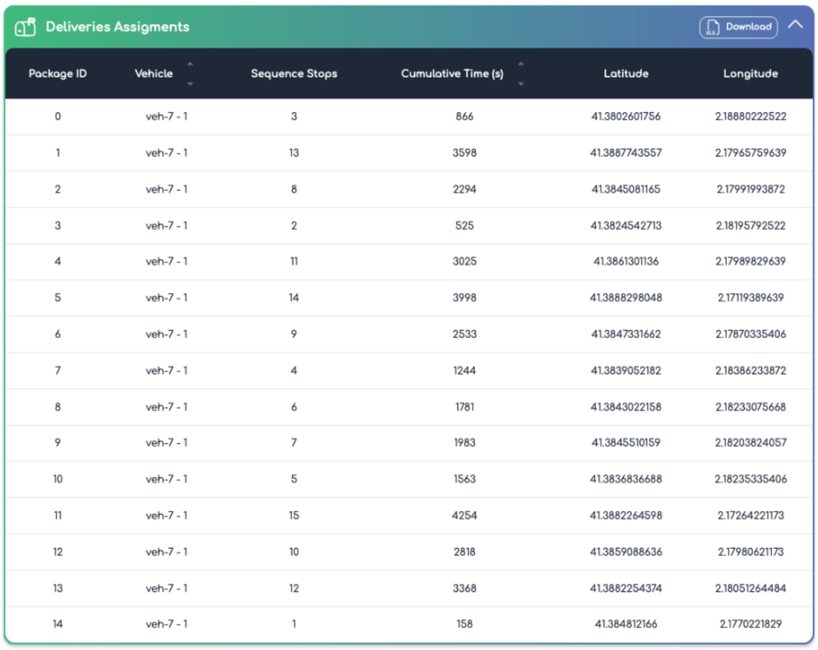
In essence, these user-generated inputs provide the parameters for the models executed in the DT (Figure 5-20) in the following order:

- Random Delivery Generation Model. This model runs only if the user does not import an Excel file at the beginning and inputs a number of deliveries to be generated randomly.
- Capacitated VRP. This model, presented in section 4.6.2, predicts the sequence of delivery locations for each e-cargo bike available in the operator's fleet.
- The *copert* model is used to calculate the CO2 emissions.
- Cost Benefit Analysis.

Comparing the planned versus the actual deliveries with e-Bikes

As explained in the previous section, this CitIQore intervention employs the VRP model to optimise the routing and scheduling of e-bikes for delivering parcels in a city context. The overall goal is to minimise costs, improve operational efficiency and reduce travel time.

This intervention was employed in the Barcelona Living Lab to plan the deliveries with Vanapedal's e-bikes in the historical city centre. Using the "Import File' functionality the Living Lab stakeholders uploaded the actual daily parcel destinations for the e-bikes. Through the CitIQore scenario execution, in the end they receive the delivery plans as outputs of the VRP model in the form of a table, which they can also export in Excel for further processing (Figure 5-21).

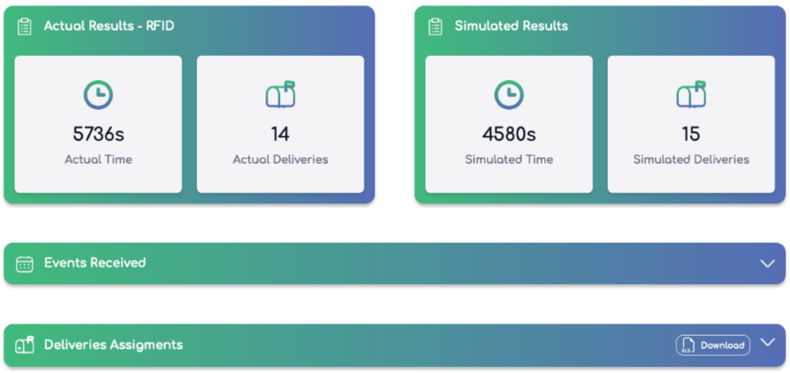

Deliveries Assigments

Download

Package ID	Vehicle	Sequence Stops	Cumulative Time (s)	Latitude	Longitude
0	veh-7 - 1	3	866	41.3802601756	2.18880222522
1	veh-7 - 1	13	3598	41.3887743557	2.17965759639
2	veh-7 - 1	8	2294	41.3845081165	2.17991993872
3	veh-7 - 1	2	525	41.3824542713	2.18195792522
4	veh-7 - 1	11	3025	41.3861301136	2.17989829639
5	veh-7 - 1	14	3998	41.3888298048	2.17119389639
6	veh-7 - 1	9	2533	41.3847331662	2.17870335406
7	veh-7 - 1	4	1244	41.3839052182	2.18386233872
8	veh-7 - 1	6	1781	41.3843022158	2.18233075668
9	veh-7 - 1	7	1983	41.3845510159	2.18203824057
10	veh-7 - 1	5	1563	41.3836836688	2.18235335406
11	veh-7 - 1	15	4254	41.3882264598	2.17264221173
12	veh-7 - 1	10	2818	41.3859088636	2.17980621173
13	veh-7 - 1	12	3368	41.3882254374	2.18051264484
14	veh-7 - 1	1	158	41.384812166	2.1770221829

FIGURE 5-21. DELIVERY ASSIGNMENTS AS OUTPUT FROM VRP

At the same time, RFID readers have been placed in the cargo bikes allowing for the real-time data capturing, sending instantaneous logs to a web server. This data is sent to the Big Data infrastructure of the Digital Twin (Figure 4-8) and is used to track the actual delivery process. Smart contracts are generated each day to track the delivery process; hence all delivery events are forwarded to the Blockchain service.


Actual Results - RFID

5736s
Actual Time

14
Actual Deliveries

Simulated Results

4580s
Simulated Time

15
Simulated Deliveries

Events Received

Deliveries Assigments

Download

FIGURE 5-22. DASHBOARD PAGE. ACTUAL (RFID) VS SIMULATED RESULTS (VRP)

When the users include the EPC (Electronic Product Code) tags, (i.e., the RFID tags) for each parcel in the Excel file they import into CitIQore, then once the Digital Twin models are executed, the actual tracked delivery events are also retrieved from the DT infrastructure.

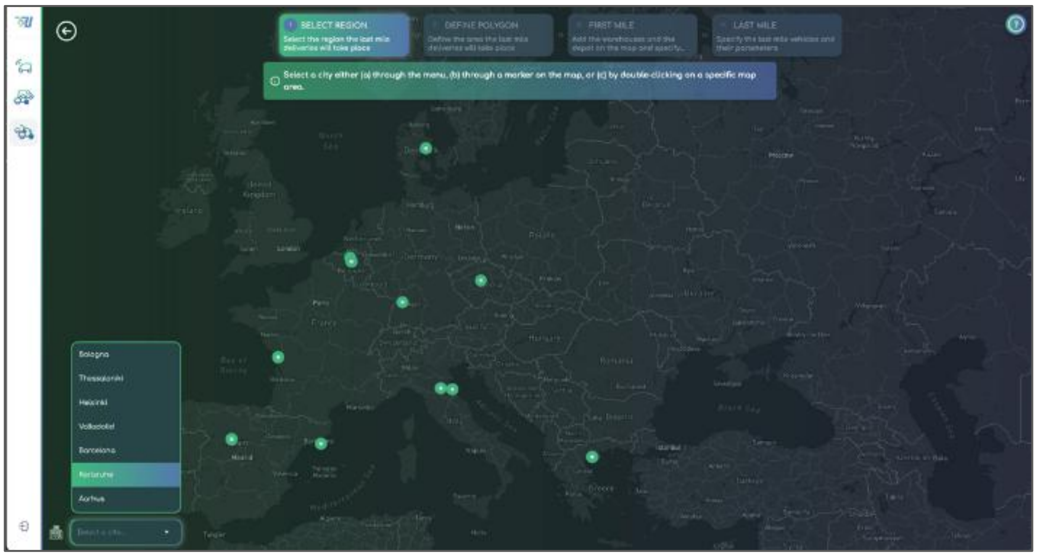
To achieve this, a backend service comes into use that retrieves all the events tracked for the smart contract of the given day. Among these events it checks all the "order in compartment" and "order delivered" or "order not delivered" events and their timestamps. It then, determines the actual delivery time for all parcels. At the same time, it calculates the total estimated time for delivery as predicted by the VRP model.

The events tracked in real time from the URBANE Platform as well as the actual times along with the predictions are returned to the frontend and presented in the CitIQore Dashboard as shown in Figure 5-22 to illustrate a comparison between the actual versus the simulated results.

A short demo video showing the overall workflow, logic, and main features — using the Last Mile Delivery with E-bikes intervention as an example — is available here: https://www.youtube.com/watch?v=mwi3Mth5YEg.

5.4 CitIQore in the context of the Follower Cities

At this point, it should be noted that the CitIQore app now supports the cities maps of the URBANE Wave 3 cities (Aarhus, Antwerp, Mechelen, Prague District 6, La Rochelle, and Ravenna). This addition enables Wave 3 stakeholders to strengthen their feasibility studies by using the app's simulation capabilities to explore how scenarios developed for the Wave 1 and 2 LLs can be adapted and applied in their own local contexts. By integrating these new city maps, the app further supports the transferability of the URBANE approach, helping partners assess the potential impact, operational needs, and practical considerations of last-mile logistics interventions in additional urban areas. For ease of use, markers for each follower city have been added to the map interface, as shown in Figure 5-23.


Select a city either (a) through the menu, (b) through a marker on the map, or (c) by double-clicking on a specific map area.

SELECT REGION
Select the region the last mile deliveries will take place
DEFINE POLYGON
Define the area the last mile deliveries will take place
FIRST MILE
Add the warehouses and the depot on the map and specify...
LAST MILE
Specify the last mile vehicles and their parameters
Bologna
Thessaloniki
Helsinki
Valladolid
Barcelona
Karlsruhe
Aarhus
Select a city...

FIGURE 5-23. MARKERS FOR FOLLOWER CITIES

6 Conclusion & Next Steps

This report describes the work undertaken in Task 3.1 and Task 3.6 of the URBANE project, more specifically the work towards the development of the URBANE Innovation Transferability Platform and Digital Twin.

It begins by outlining the functional and non-functional requirements for the URBANE Digital Twin Platform, as well as the conceptual architecture of the Innovation Transferability Platform. The report then examines the technical architecture of the Platform, the underlying infrastructure, and the integration of LL data and models.

Several key implementation steps of Task 3.6 are detailed, including the design and implementation of the Platform's data pipeline, which enables the real-time ingestion of data from the project's LLs. This real-time capability supports the broader digitalisation of transport and logistics by providing a robust foundation for continuous, data-driven insights and operational improvements.

A central aspect of the work was the integration of the URBANE models, including their metadata into the Digital Twin Platform. Since these technical solutions developed in WP3 are applied within the project's LLs, their successful integration ensures transferability and applicability to similar use cases across different contexts. By enabling stakeholders to measure and evaluate different last mile logistic scenarios and their impact on reducing CO2 emissions, these models play a vital role in supporting cities and logistics operators to test, plan, and refine sustainable urban logistics strategies.

Another important outcome is the development of CitIQore — a domain-specific application that provides a user-friendly interface for interacting with the URBANE Digital Twin Platform. CitIQore empowers city authorities and logistics industry stakeholders to explore "what-if" scenarios and assess alternative decision pathways for urban logistics operations in a clear and accessible way.

In conclusion, the work presented in this report contributes not only to the development of the URBANE Innovation Transferability Platform but also to the broader digitalisation of the transport and logistics sector. By enabling stakeholders to measure, model, and evaluate different scenarios for optimising last-mile delivery operations, the URBANE Digital Twin and Innovation Transferability platform provides valuable tools for planning interventions that can help reduce CO_2 emissions and support more sustainable urban freight solutions. The developed solutions are designed to be replicable and transferable across different city contexts, helping stakeholders in Europe and beyond adopt data-driven approaches for more efficient, low-carbon, and resilient urban logistics systems.

7 Bibliography

[1] E. Commission, “Workshop - Local Digital Twins Technology,” [Online]. Available: https://digital-strategy.ec.europa.eu/en/events/workshop-local-digital-twins-technology.

[2] C. C. M. S. D. L. Azad M. Madni, “Leveraging Digital Twin Technology in Model-Based Systems Engineering,” *Systems,* vol. 7, no. 7, 2019.

[3] E. T. a. R. G. Thompson, “Modeling City Logistics,” *Transportation Research Record: Journal of the Transportation Research Board,* vol. 1790, no. 1, pp. 45-51, 2002.

[4] W. J. G. P. a. D. S. P. Antosz, “Informing Agent-Based Models of Social Innovation Uptake,” pp. 105-117, 2021.

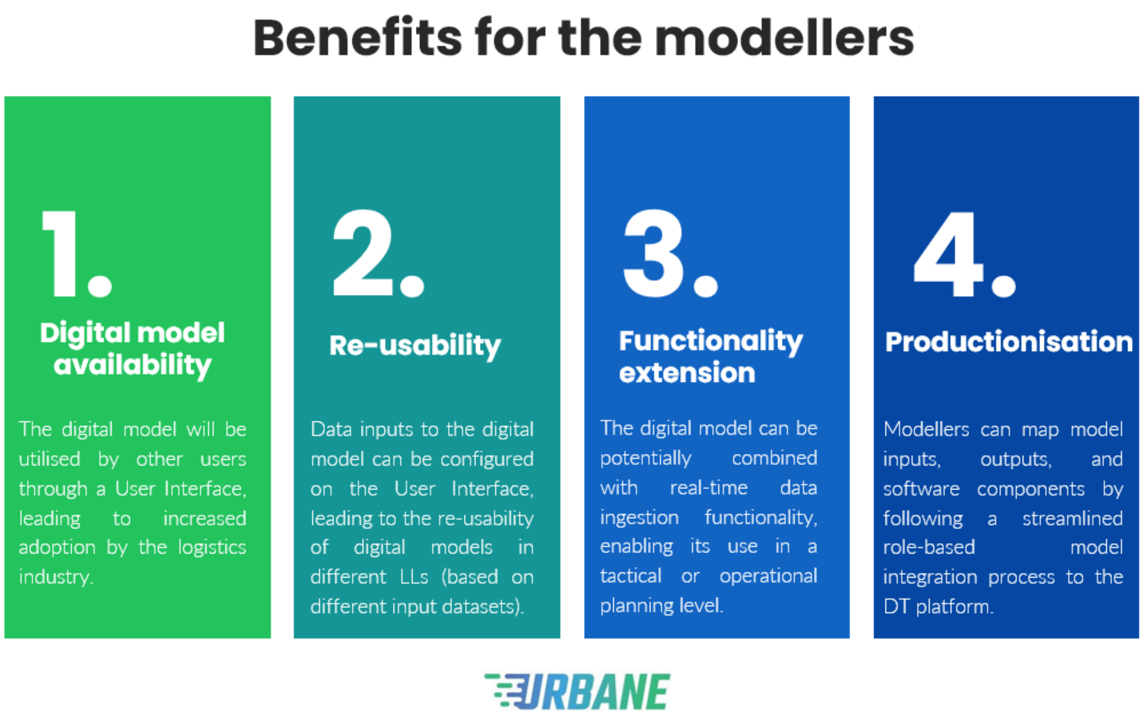
[5] R. T. a. I. Kourounioti, “LEAD Deliverable D1.2: Knowledge Base Reference Models,” 2022.

[6] “Planet project: Progress towards Federated Logistics through the Integration of TEN-T into A Global Trade Network,” European Commission, May 2023. [Online]. Available: https://www.planetproject.eu.

Annex I – Integration Process and Guidelines

Digital model integration with the Digital Twin Platform

Digital last-mile network models produced within URBANE's WP2 & WP3 can be integrated into the URBANE Digital Twin (DT) platform, enabling their execution by Living Lab stakeholders (i.e., decision-making consultants, operations consultants) through an easy-to-use User Interface (UI). The DT Platform can offer real-time data ingestion and combine real-time data feeds with the digital models if the LL Use cases and models require it.



Benefits for the modellers

1. Digital model availability

The digital model will be utilised by other users through a User Interface, leading to increased adoption by the logistics industry.

2. Re-usability

Data inputs to the digital model can be configured on the User Interface, leading to the re-usability of digital models in different LLs (based on different input datasets).

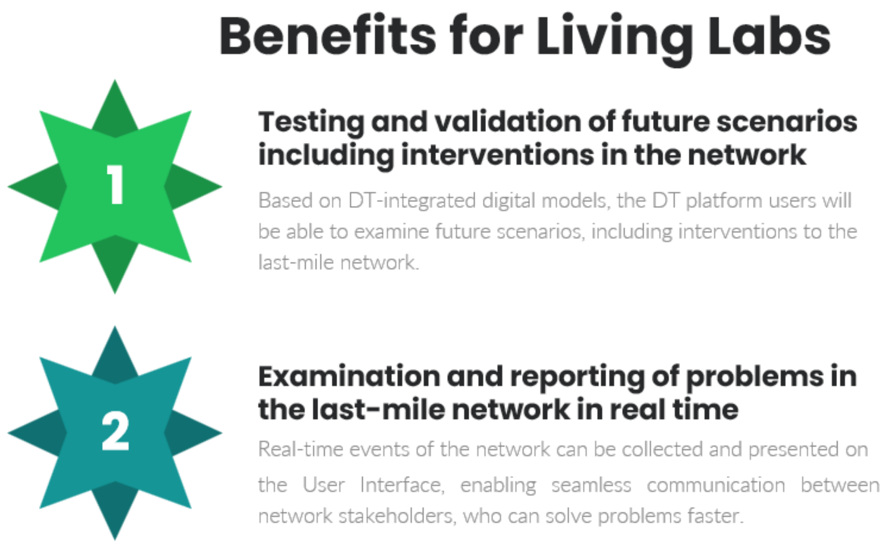
3. Functionality extension

The digital model can be potentially combined with real-time data ingestion functionality, enabling its use in a tactical or operational planning level.

4. Productionisation

Modellers can map model inputs, outputs, and software components by following a streamlined role-based model integration process to the DT platform.

URBANE



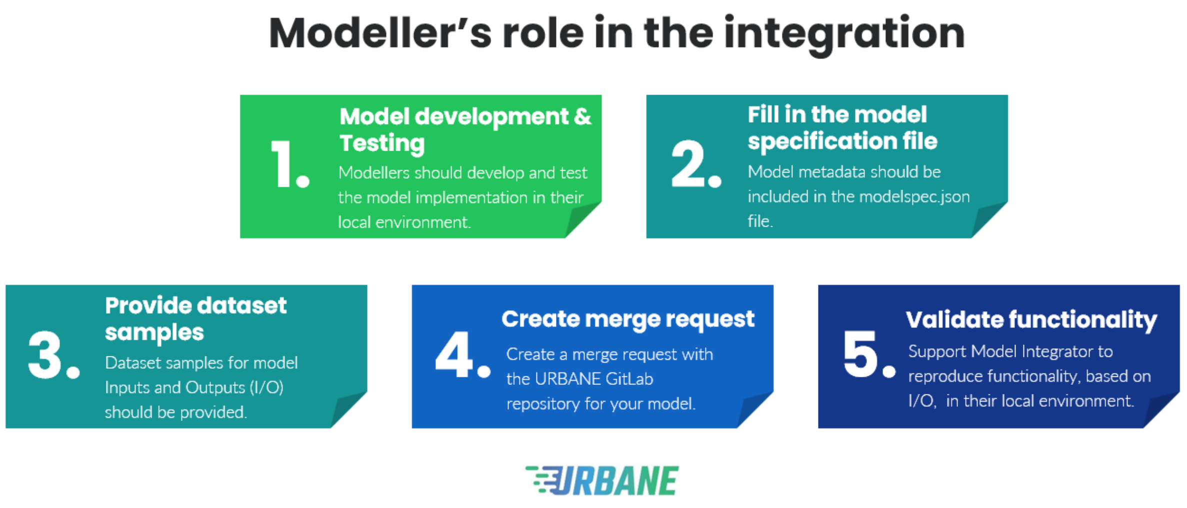
Benefits for Living Labs

1 Testing and validation of future scenarios including interventions in the network

Based on DT-integrated digital models, the DT platform users will be able to examine future scenarios, including interventions to the last-mile network.

2 Examination and reporting of problems in the last-mile network in real time

Real-time events of the network can be collected and presented on the User Interface, enabling seamless communication between network stakeholders, who can solve problems faster.



Modeller's role in the integration

1. Model development & Testing
Modellers should develop and test the model implementation in their local environment.

2. Fill in the model specification file
Model metadata should be included in the modelspec.json file.

3. Provide dataset samples
Dataset samples for model Inputs and Outputs (I/O) should be provided.

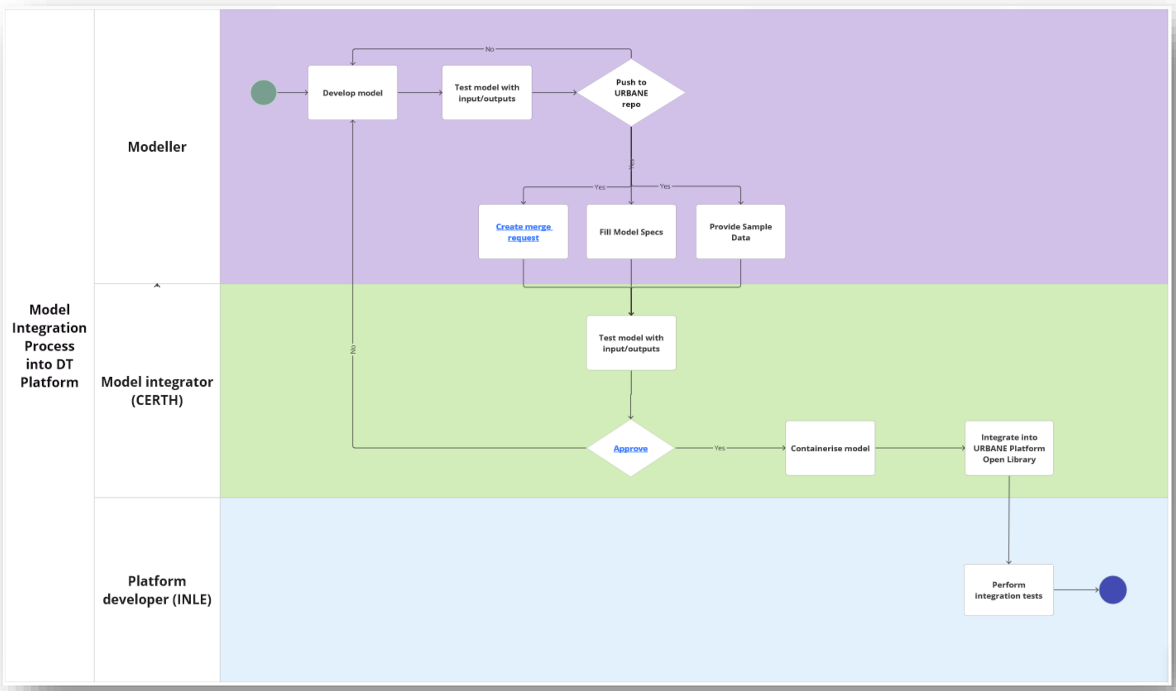
4. Create merge request
Create a merge request with the URBANE GitLab repository for your model.

5. Validate functionality
Support Model Integrator to reproduce functionality, based on I/O, in their local environment.

URBANE

General description integration process

For the successful integration of the models in the DT platform, the URBANE project shall follow a specific integration process that identifies three roles: *i)* the modellers, *ii)* the model integrator, *iii)* the DT platform developer. The integration process can be found in the following figure:


Model Integration Process into DT Platform
Modeller
Model integrator (CERTH)
Platform developer (INLE)
Develop model
Test model with input/outputs
Push to URBANE repo
No
Yes
Yes
Yes
Create merge request
Fill Model Specs
Provide Sample Data
Test model with input/outputs
No
Approve
Yes
Containerise model
Integrate into URBANE Platform Open Library
Perform integration tests

Each role within the integration process is assigned different responsibilities. Starting with the modellers their focus is the development of the project's models. However, for the models to be transferable and easy-to-integrate within the DT platform, the code needs to be (a) already tested by the modeller and (b) configured properly. The platform treats the models as black boxes yet expects clearly defined input and output parameters. Therefore, the model inputs are to be exposed as either Environment (ENV) or

Command Line Input (CLI) arguments -i.e., not hardcoded within the model's code. In this context, the Platform offers a Model Specification file in which the modellers are called to fill all the metadata (including input parameters and output parameters) of their models (examples can be found here: modelspec.json for Python models, modelspec.json for R models). Samples for the models have been developed by the Inlecom team to support modellers in the model development and integration process which can be found here:

https://gitlab.com/urbane-horizon-europe/model-library/samples

If possible, the modellers shall provide sample data for further testing by the model integrator role as this can further assist the integration process. Finally, the modellers are called to upload their code on the URBANE Models Library (Gitlab repository), and create a merge request on the GitLab repository.

Based on these inputs from the modellers, the Digital model integrator is responsible for the URBANE Model Library (and respective GitLab repository). The role of the Model integrator is to guarantee that the models run in any environment following the successful run and testing by the modellers in their own local environments. After this examination, the digital model integrator is expected to containerise the digital model using Docker. The containerised digital model, model specification file and data assets shall be placed in the Gitlab repository for each model. Changes in the digital model code in the initial branch should be merged (included) with the second branch (that includes the Docker image). Throughout this process the Model integrator will collaborate with the modellers.

Finally, the DT platform developer is in charge of the onboarding of containerised digital models within the platform along with the customized development of the DT platform to serve the models and Living Lab Use cases. This includes all the software components responsible for the digital model execution, real-time data ingestion, results presentation, and visualisation. Additional features, except for the ones discussed in this document, may be added based on requests from LLs.