

INTEGRATED PLATFORM FOR ASSESSING SELF-ORGANISING RAILWAY OPERATIONS

Grant Agreement N°:	875022
Project Acronym:	SORTEDMOBILITY
Project Title:	Self-Organized Rail Traffic for the Evolution of Decentralized MOBILITY
Funding scheme:	Horizon 2020 ERA-NET Cofund
Project start:	1 June 2021
Project duration:	3 Years
Work package no.:	4
Deliverable no.:	2
Status/date of document:	Final, 11/01/2024
Due date of document:	30/11/2023
Lead partner for this document:	TU Delft
Project website:	www.sortedmobility.eu

Dissemination Level		
PU	Public	
RE	Restricted to a group specified by the consortium and funding agencies	
CO	Confidential, only for members of the consortium and funding agencies	

Revision control / involved partners

Following table gives an overview on elaboration and processed changes of the document:

Revision	Date	Name / Company short name	Changes
1	15/09/2023	TU Delft	Deliverable structure with description of content for each section
2	10/12/2023	TU Delft	First final draft issued including contributions of UNI EIFFEL, ISTC-CNR, DTU and TU Delft.
3	20/12/2023	TU Delft	Content revision of all document sections
4	08-01-2024	UNI-EIFFEL	Feedback provided on document content
5	08-01-2024	TU Delft	Final document revised based on received feedback from partners

Following project partners have been involved in the elaboration of this document:

Partner No.	Company short name	Involved experts
1	univEiffel	Paola Pellegrini
2	SNCF	
3	DTU	Carlos Avezedo da Lima, Georges Sfeir
4	BDK	
5	ISTC-CNR	Fabio Oddi, Vito Trianni
6	RFI	
7	TUDelft	Egidio Quaglietta, Kemal Keskin, Konstantinos Rigos

Executive Summary

The primary goal of D4.2 is to describe the flexible simulation platform for evaluating the impact of self-organizing rail traffic management algorithms on train service performance and customer satisfaction. A Recap is provided on the algorithms for travel demand and self-organizing real-time traffic management which are used in the assessment platform. A detailed description of the input/output modules and functionalities of the microscopic rail traffic simulation tool EGTRAIN is reported together with newly built functions specifically made to represent and assess self-organizing traffic operations. A scheme of the simulation platform architecture is specified including the description of data interfaces and format which are adopted to dynamically exchange information among the different interacting modules and algorithms. A web-based AMQP architecture based on the Zero-MQ interface is discussed to provide scalability of intermodular information exchange. Building such a data interface paradigm, a modular and scalable architecture is proposed which allows flexibility to interchange modules and algorithms as well as independency from the underlying rail traffic simulation tool.

Table of contents

INTEGRATED PLATFORM FOR ASSESSING SELF-ORGANISING RAILWAY OPERATIONS	1
1 INTRODUCTION	7
2 A SIMULATION FRAMEWORK FOR ASSESSING SELF-ORGANISING RAILWAY OPERATIONS	8
2.1 The EGTRAIN rail traffic simulation platform	8
2.1.1 Existing Simulation Architecture	8
2.1.2 Infrastructure module	10
2.1.3 Rolling Stock module	12
2.1.4 Signalling system module	15
2.1.5 Timetable module	20
2.1.6 Outputs of simulation	22
2.2 Self-organising operational principles and KPIs	24
2.3 Demand models for route choice	25
2.3.1 Model framework	26
2.3.2 Path set generation	27
2.3.3 Route choice model	28
2.3.4 Route choice model parameters	29
2.4 Self-organising rail traffic management algorithms	30
3 SIMULATION INTERFACE WITH DEMAND MODELS	32
3.1 Passenger flow simulation module in EGTRAIN	32
3.2 Interaction with daily activity schedule and route choice	35
3.3 Passenger-dependent train dwell time estimation	35
4 SIMULATION INTERFACE WITH ALGORITHMS FOR SELF-ORGANIZING TRAFFIC MANAGEMENT	38
4.1 Transport State Monitoring module in EGTRAIN	38
4.2 Traffic management module in EGTRAIN	41
4.3 Interaction with self-organising traffic management algorithms	42
4.4 Traffic management communication interface	42
5 CONCLUSIONS	44
6 REFERENCES	45

Table of figures

Figure 1. Structure of the synchronous microscopic model	9
Figure 2. Link-orientated graph model of a simple double-track real network.	10
Figure 3. Direction convention for the links of the graph model	11
Figure 4. Input text files of the infrastructure module containing the database of respectively node and link attributes of the “even” track	12
Figure 5. Input text file of train physical characteristics.	14
Figure 6. Input text file of train mechanical attributes of the traction unit.	15
Figure 7. Graph model network with signalling layout.....	16
Figure 8. Input database for the definition of block section layout.	17
Figure 9. Sample interlocking system with diverging position (Case A) and non-diverging position (Case B)	18
Figure 10. Input database for the definition of the layout of interlocking elements.....	19
Figure 11. Graph model of the network with signalling and interlocking layout.....	20
Figure 12. Input database for the definition of the cyclic timetable.....	21
Figure 13. Text file containing the database of simulation outputs relative to a certain train for a single replication.....	23
Figure 14. Text file containing the database of train arrival times at each station for a single replication.....	24
Figure 15. UML Class representation of Passenger module.	34
Figure 16. The interaction with daily activity schedule and route choice	35
Figure 17. Software interfaces.....	38
Figure 18. TSM.xml file without passenger information	39
Figure 19. TSM.xml file with passenger information	40
Figure 20. RTTP.xml file	41
Figure 21. Flow diagram of ZeroMQ.	43



Tables

Table 1. Model results	29
Table 2. Parameter estimates.....	29
Table 3. Default parameters of Gibson dwell time model.....	36

1 INTRODUCTION

SORTEDMOBILITY aims to revolutionize urban and interurban transportation by establishing rail transportation as a fundamental element of mobility. The report focuses on enhancing the EGTRAIN simulator to simulate self-organizing trains based on operational principles and algorithms developed in the project.

The research in this report has the objective of developing a flexible and scalable simulation platform to assess the impact of self-organising rail traffic management algorithms on both train service performance (such as punctuality, total delay) and customer satisfaction. The different modules of the developed simulation architecture are described in terms of functionalities as well as input and output data. A description of the input/output interaction flow among the different components is given together with corresponding data formats. An overview is given on standard data format used in SORTEDMOBILITY which make the simulation framework flexible, allowing independence, therefore interchangeability of the underlying traffic simulation environment. The report focuses on specific developments of the EGTRAIN simulation software, however the architecture can be applicable to any other microscopic rail traffic simulation tool, provided they feature similar APIs for interacting with travel demand and traffic management algorithms.

The report is organized as follows. Chapter 2 provides a detailed explanation of the simulation framework for assessing self-organizing railway operations. Chapter 3 presents passenger flow simulation module of EGTRAIN and various demand models. New modules of EGTRAIN, Transport State Monitoring module and Traffic management module and interaction with self-organising traffic management algorithms has been introduced in Chapter 4. Chapter 5 includes conclusions.

2 A SIMULATION FRAMEWORK FOR ASSESSING SELF-ORGANISING RAILWAY OPERATIONS

2.1 The EGTRAIN rail traffic simulation platform

Railway simulation models are crucial for planning and design. They help assessing how changes can impact railway performance. These models can be used for both designing infrastructure and operations (like timetables) and managing real-time train operations. It is essential for these models to provide accurate results quickly, especially during real-time adjustments or when testing various scenarios.

Although detailed models, called microscopic models, offer accurate train dynamic analysis, they often are computationally expensive, especially for large networks. This is due to their detailed input requirements and lack of flexibility with commercial software. As a result, people often use simpler, less detailed models (e.g. macroscopic or mesoscopic), especially when testing many scenarios or dealing with larger networks. However, those less detailed models can sometimes give results that aren't accurate enough for congested and disturbed traffic situations.

EGTRAIN, written in C++ using object-oriented programming, is flexible and features APIs to customize functionalities and interact with external tools. It is designed to run faster on computers with multiple cores and provides detailed representation of all railway components, to provide accurate description of train dynamics. Users can customize this model for any situation, making it versatile and adaptable.

2.1.1 Existing Simulation Architecture

The EGTRAIN simulator has an open modular structure, allowing manipulation of inner functions without altering the original source code. EGTRAIN adapts to user requirements and is able to interact with external programs and mathematical structures, either directly or indirectly via an API module. It also allows for the modification or addition of core simulation functions (Quaglietta, 2011).

EGTRAIN was designed to reduce the effort for initialising input data for microscopic representation of real case studies and allow customisation of simulation functions and modules. EGTRAIN is structured into four main modules, each

representing a specific railway system component (infrastructure, rolling stock, signalling system, timetable). Users can customize input datasets for each module through external files, eliminating the need to modify the source code for different case studies.

Microscopic infrastructure models can be inefficient due to their extensive input data requirements. However, modern multi-core computers and multi-threading programming have been leveraged to enhance efficiency. Implementing a parallel architecture required significant effort but resulted in faster simulations, especially for larger networks and multi-core systems, without compromising accuracy. EGTRAIN has the advantage of using parallel architecture and its computational performance is reasonable, even when simulating large networks or conducting extensive stochastic model evaluations.

The Figure 1 illustrates the structure of the existing synchronous microscopic simulation model. It reveals that the input data is managed across four interconnected modules:

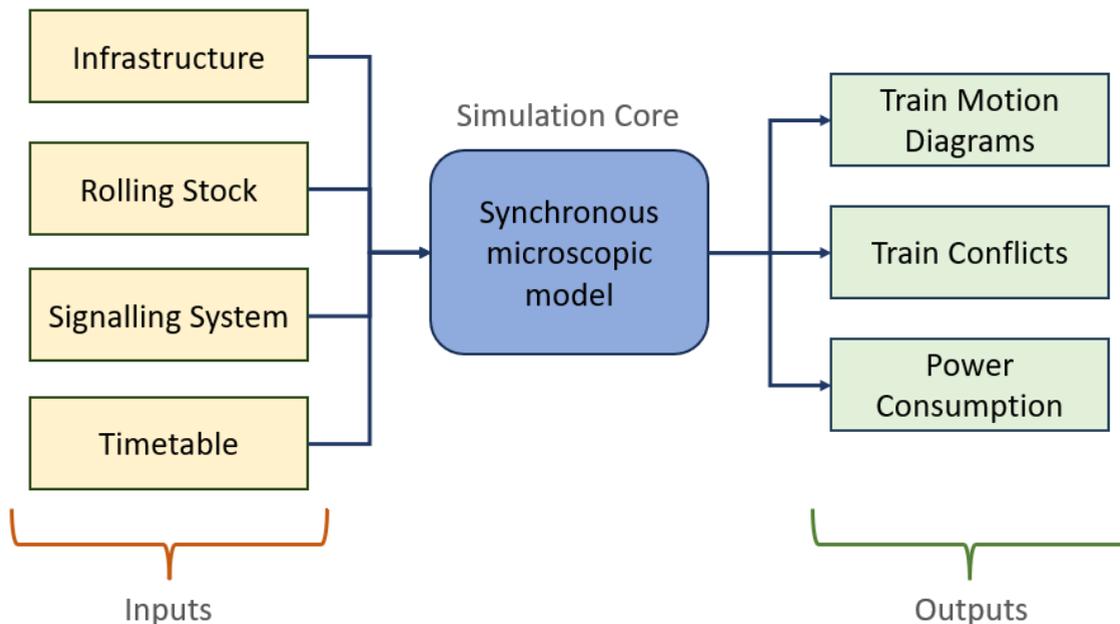


Figure 1. Structure of the synchronous microscopic model

The simulation core operates synchronously, replicating network events in real-time order for each time increment within the total simulation period. It determines the status values, like line-side signal aspects or rail vehicle positions, for all components at each moment. This model produces various outputs, such as

train motion charts, conflict configurations, energy consumption graphs, and statistics on train punctuality and delays. Subsequent sections will go deeper into the model's four main modules.

2.1.2 Infrastructure module

The railway network is represented as a link-oriented graph, where nodes store information about station, signal, and switch positions. Links specify rail track characteristics like radii, gradients, and speed limits. Creating this graph involved defining node and link objects. Node attributes include a unique ID and spatial coordinates (X and Y in km). This module doesn't consider "signal" nodes; it considers a node whenever a link attribute changes. Link-oriented graphs assume homogeneous link characteristics, so when any attribute changes (e.g., speed limit or gradient), a new homogeneous link and corresponding node are created. Input data must provide attributes for these nodes. Additionally, station nodes indicating platform positions are required. Figure 2 illustrates how a double-track railway network is represented in this graph model.

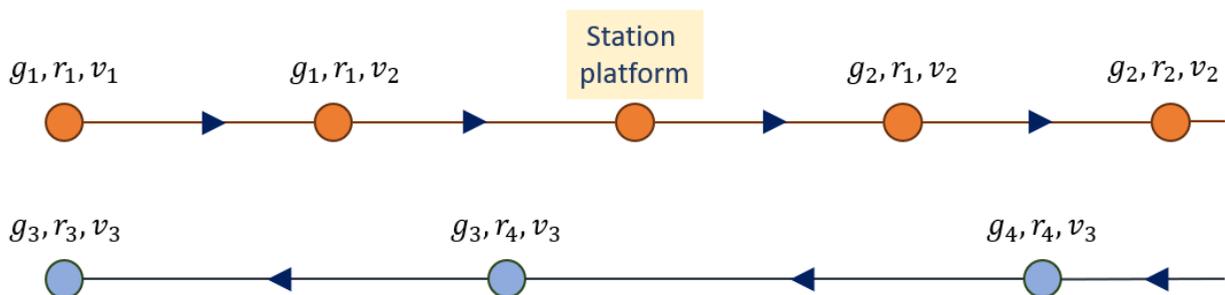


Figure 2. Link-orientated graph model of a simple double-track real network.

For graph links, a new class was created with attributes such as; Link ID, Start and End node IDs, Curvature radius, Gradient, and Speed Limit.

The link length is derived using the Pythagorean theorem, based on the provided positions of the start and end nodes. Link direction, vital for train movement, is set using a Cartesian reference system (see Figure 3). Links which are in the same direction as the positive X axis take a value of 1, those in the opposite direction take the value -1, and bi-directional links receive a value of 0.

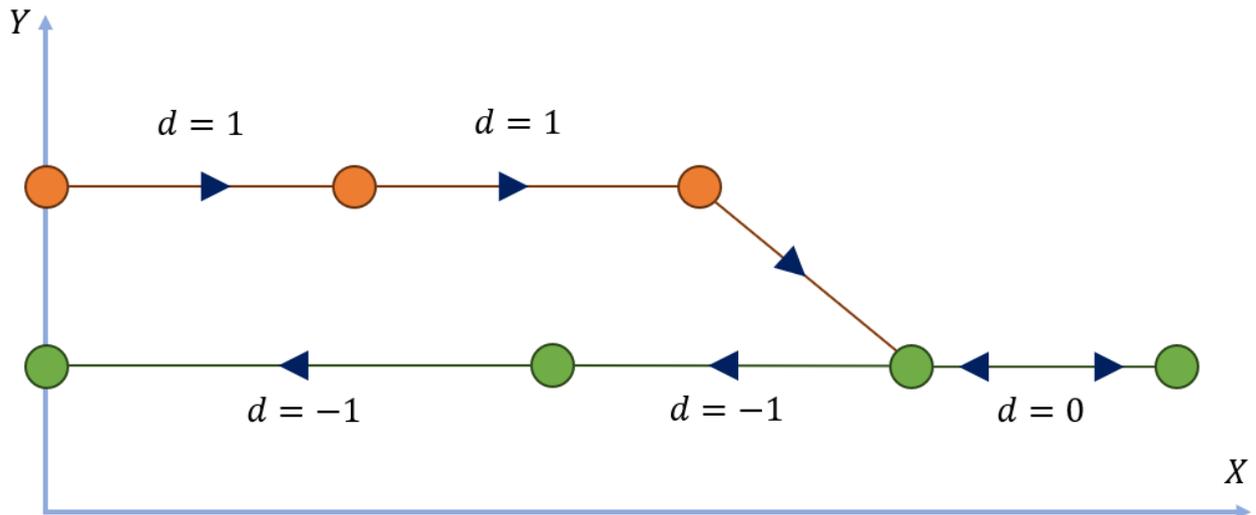


Figure 3. Direction convention for the links of the graph model

Node and link characteristics need to be organized in a text file following a database format. Figure 4 provides a visual representation of this format, where the columns, define the attributes of elements such as nodes or links. In the specific example depicted in Figure 4, a double-track network is used as the default setting in the program. It's important to note that link direction is not considered in the link database when using the default mode. However, if the default mode is bypassed, an additional field (direction) must be added to the link database to specify the direction of all links (1, 0, or -1).

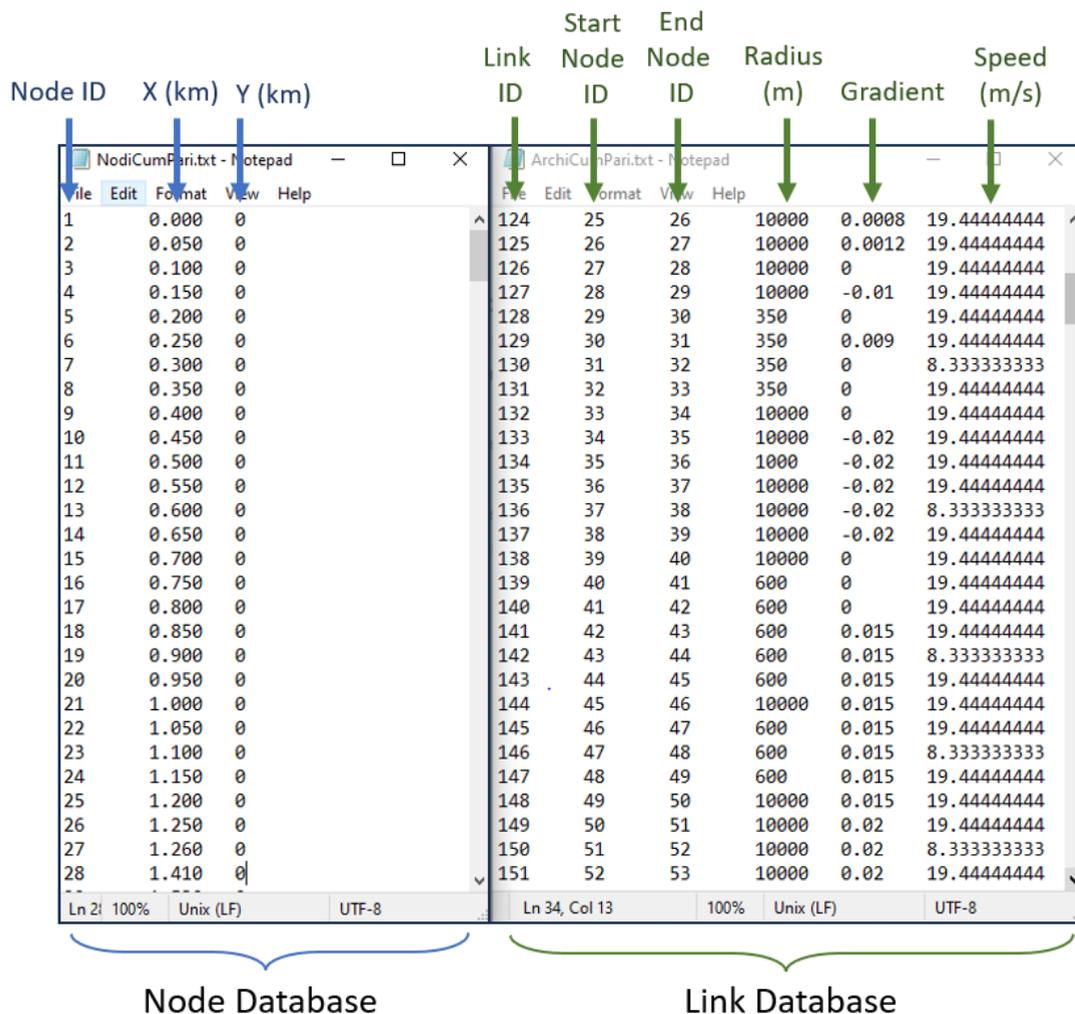


Figure 4. Input text files of the infrastructure module containing the database of respectively node and link attributes of the “even” track

2.1.3 Rolling Stock module

To accurately capture the dynamic behaviour of rail vehicles, including transient phases like acceleration and deceleration, a comprehensive description of all physical and mechanical vehicle characteristics is required. This is achieved by defining a 'vehicle' object and specifying its attributes, such as vehicle length, weight, number of coaches, deceleration rate, and so on. Additionally, coefficients related to the locomotive's tractive effort-speed curve must be provided. The specific vehicle attributes that users need to input such as Train ID, Mass of the traction unit m_T (kg), Mass of a single wagon m_{Wi} (kg), Number of wagons n_W , Maximum speed of the traction unit v_{max} (m/s), Service deceleration rate b_s (m/s²)

), Cross-sectional area of the vehicles, A_f (m²), Southoff formulae coefficient c_b , Jerk rate J (m/s³), Total length of the rail vehicle L (m) are defined.

This set of attributes serves to initialize both physical train characteristics and parameters of motion resistance equations. These attributes are essential for integrating Newton's motion formula and, consequently, simulating train movements on the track. Notably, the traction unit's weight, m_T , is also employed to determine traction unit resistances. Traction unit resistances are calculated using the Italian FS formula which is given as follows:

$$R_{TR}(v) = 4.2 \cdot m_T \cdot g + 0.72 \cdot v^2$$

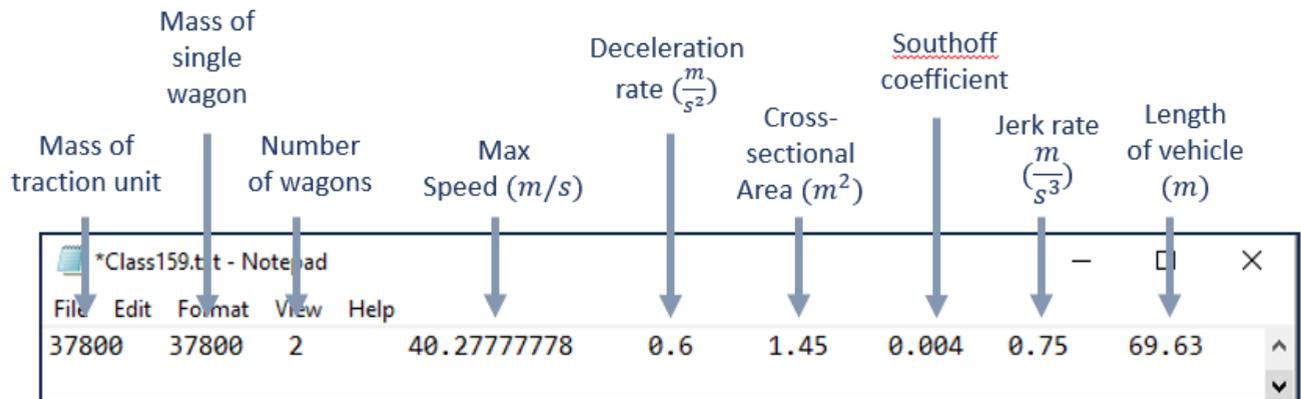
Motion resistances of passenger trains due to air viscosity can be calculated using Southoff formulae which is presented as follows:

$$R_w(v) = (1.9 + c_b \cdot v) \cdot \frac{g \cdot m_w}{1000} + 4.7 \cdot (n_w + 2.7) \cdot A_f \cdot \left(\frac{v + 15}{10}\right)^2$$

The total wagon mass (m_w) in the formula is simply calculated as the product of the number of wagons (n_w) and the mass of a single wagon (m_{wi}), assuming all wagons are identical. Those resistance formulas are used as default in the EGTRAIN simulation functions, however other air drag resistance models such as Davis' (1926) can be devised.

Train length (L) is crucial for accurately determining the occupation times of each block section. Clearing and release times depend on both the type of signaling system in place and the length of the trains. Therefore, train length plays a significant role in accurately estimating network capacity.

The data format of the input file to be provided for this module can be found in Figure 5.



Mass of traction unit	Mass of single wagon	Number of wagons	Max Speed (m/s)	Deceleration rate ($\frac{m}{s^2}$)	Cross-sectional Area (m^2)	Southoff coefficient	Jerk rate ($\frac{m}{s^3}$)	Length of vehicle (m)
37800	37800	2	40.27777778	0.6	1.45	0.004	0.75	69.63

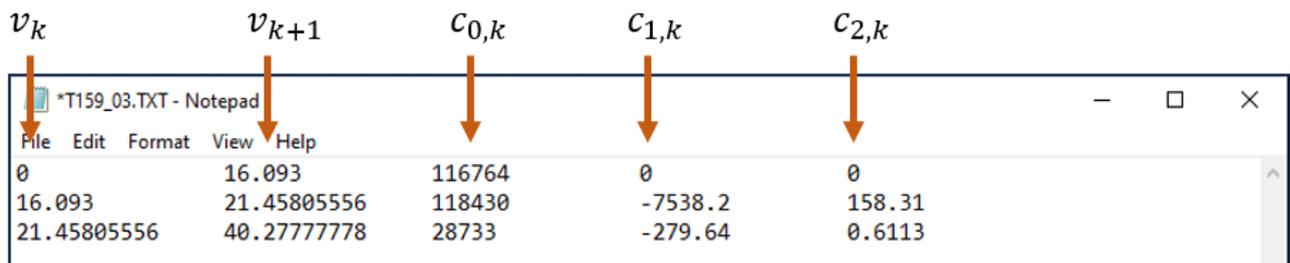
Figure 5. Input text file of train physical characteristics.

Regarding the traction unit's mechanical characteristics, all coefficients of the locomotive's "tractive effort-speed" curve must be provided as input data, as defined by the following formulation.

$$F_{Ti}(v) = c_{0,k} + c_{1,k} \cdot v + c_{2,k} \cdot v^2, \quad v_k < v \leq v_{k+1}$$

This equation calculates the tractive force generated between the wheel rim and the rails at the current running speed, v .

The tractive effort versus speed curve is demonstrated as a set of parabolas. In order to specify each parabola it is required to be entered some mechanical attributes such as; lower bound of the k^{th} speed domain, v_k (m/s), upper bound of the k^{th} speed domain, v_{k+1} (m/s), parameters of the k^{th} curve, $c_{0,k}$ (N), $c_{1,k}$ (Ns/m), $c_{2,k}$ (Ns²/m²). To incorporate train mechanical attributes into this module, the data must be organized in a database format within a text file. Specifically, the k^{th} record of this database will include both domain speed bounds and coefficients associated with the k^{th} parabolic curve that represents the mechanical curve. The database fields (i.e., columns) contain values for domain bounds and curve coefficients. The data format of the input file to be provided for this mechanical attributes of the traction unit can be found in Figure 6.



	v_k	v_{k+1}	$C_{0,k}$	$C_{1,k}$	$C_{2,k}$
0	16.093	116764	0	0	
16.093	21.45805556	118430	-7538.2	158.31	
21.45805556	40.27777778	28733	-279.64	0.6113	

Figure 6. Input text file of train mechanical attributes of the traction unit.

Additionally, the user has the option to define the adhesion coefficient μ between the wheel rim and the rail. This parameter introduces a distinct upper limit for tractive effort F , which in this instance is no longer governed by mechanical characteristics but rather by adhesion phenomena. Consequently, the tractive effort F employed during the integration of Newton's motion formula will always adhere to the following condition: $F \leq \mu \cdot G$, where G represents the total train weight.

Users can simulate different types of rail vehicles by providing the necessary physical and mechanical attributes for each type. The number of trains and their departure headway can be set directly in the program code, allowing for the creation of a cyclic timetable. For irregular train departures, the timetable module provides the option to specify individual departure times.

2.1.4 Signalling system module

This module simulates how signalling equipment works and interacts with trains and other parts of the rail network. Signalling systems are responsible for safely controlling train movements on both main tracks and in stations and shunting yards. This is done using signals and other warning devices that only allow trains to move when it is safe to do so. However, this module needs to know what type of signalling system is used on the network, the speed code pattern, the layout of the block sections, and the positions and features of the switches. This module includes three different types of signalling systems: 1) Coded track circuit system (specifically the Italian version B.A.C.C.) 2) ETCS level 1 system 3) ETCS level 2 system.

To accurately simulate the behaviour of different signalling systems and their interactions with other parts of the network, this module provides a specific class that allows users to specify the type of signalling system, speed codes, and delay times. Users must also define the positions of signals and/or balises. The

signalling layout is then automatically created, and the resulting network graph is composed of both physical and signal nodes.

The module supports three different types of signalling systems: coded track circuit, ETCS level 1, and ETCS level 2. For coded track circuits, users can specify the speed codes for different signal aspects. For ETCS systems, users can specify the delay time for signalling equipment to communicate the signal aspect to the train.

Once the signalling system is specified, users must define the positions of signals and/or balises. This is necessary to determine the layout of the block sections. Once the block section lengths are specified, the corresponding signalling layout is automatically created.

The signalling layout is then overlaid on the physical infrastructure network. This is done by enriching the graph model representing the infrastructure network with additional signal nodes positioned at the block section joints as defined by the user. The resulting network graph is therefore composed of both physical (e.g., stations) and signal nodes (e.g., line-side signals) as seen in Figure 7.

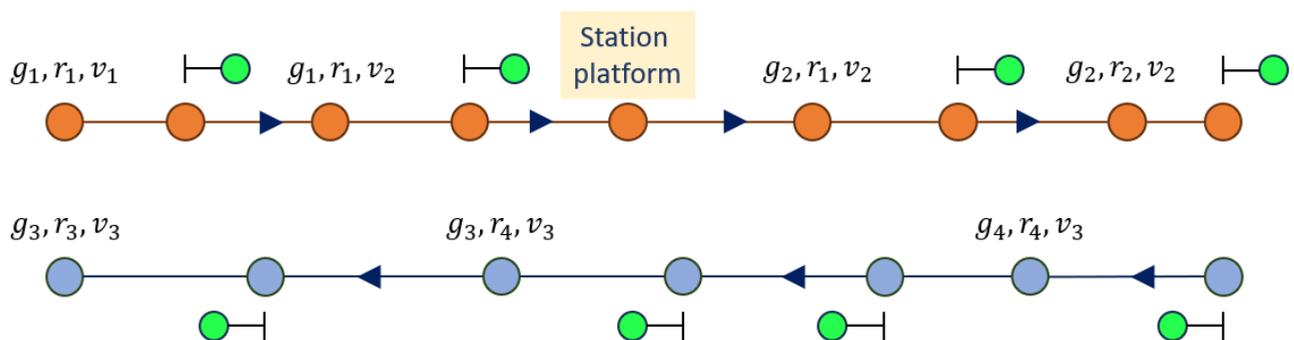


Figure 7. Graph model network with signalling layout.

Block section length defines the distance between two consecutive main signals. Users can choose between two alternatives: equi-block section layout (all block sections have the same length) or non-equal block section layout (block sections can have different lengths).

To implement equi-block section layout, users simply need to specify the length in kilometers in the program code. All block sections in station areas will be created according to the equi-block layout. On open tracks, block sections may have different lengths than the equi-block length, because the ratio of the inter-station

track length to the equi-block length is not always an integer. In these cases, the remaining length must be partitioned evenly among all the sections that compose the open track.

To implement the second block section layout, users must specify the length of each block section. This can be done by storing the block section data in a database, which can be saved as a text file. The attributes such as 'Block section index' and 'Block section length' must be defined for each block section.

Block index	Block length
0	0.064
1	0.300
2	0.115

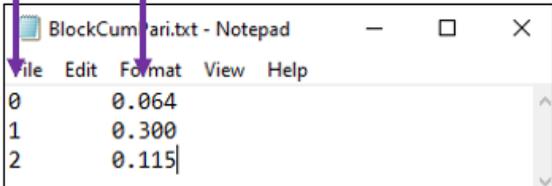


Figure 8. Input database for the definition of block section layout.

For networks with double-track layout, this data must be defined for both directions. Figure 8 shows an example of a text file containing the attributes of block section layouts.

Interlocking systems ensure safe train movements by allowing a train to proceed on a switch in a diverging position only if safe conditions are met. To achieve this, all potential conflicting trains are given a restricted aspect. To accurately model the behaviour of interlocking systems, it is necessary to define which block sections show a proceed aspect and which are blocked for each switch position.

The example in Figure 9, case A, shows that when switch SW1 is locked in the diverging position to allow train T1 to move, main signals of block section BS3 show a red aspect to prevent train T2 from moving in the opposite direction. This way, train T1 can enter block section BS2 while block section BS3 is blocked to train T2.

Conversely, when switch SW1 is not locked in the diverging position (case B), train T2 can enter block section BS3 while train T1 must respect the restricted aspect given to main signals of block section BS2.

To accurately model the behaviour of interlocking systems and their dependence on signals and moving elements (i.e., switches), it is necessary to define which

block section shows a proceed aspect and which is blocked for each switch position.

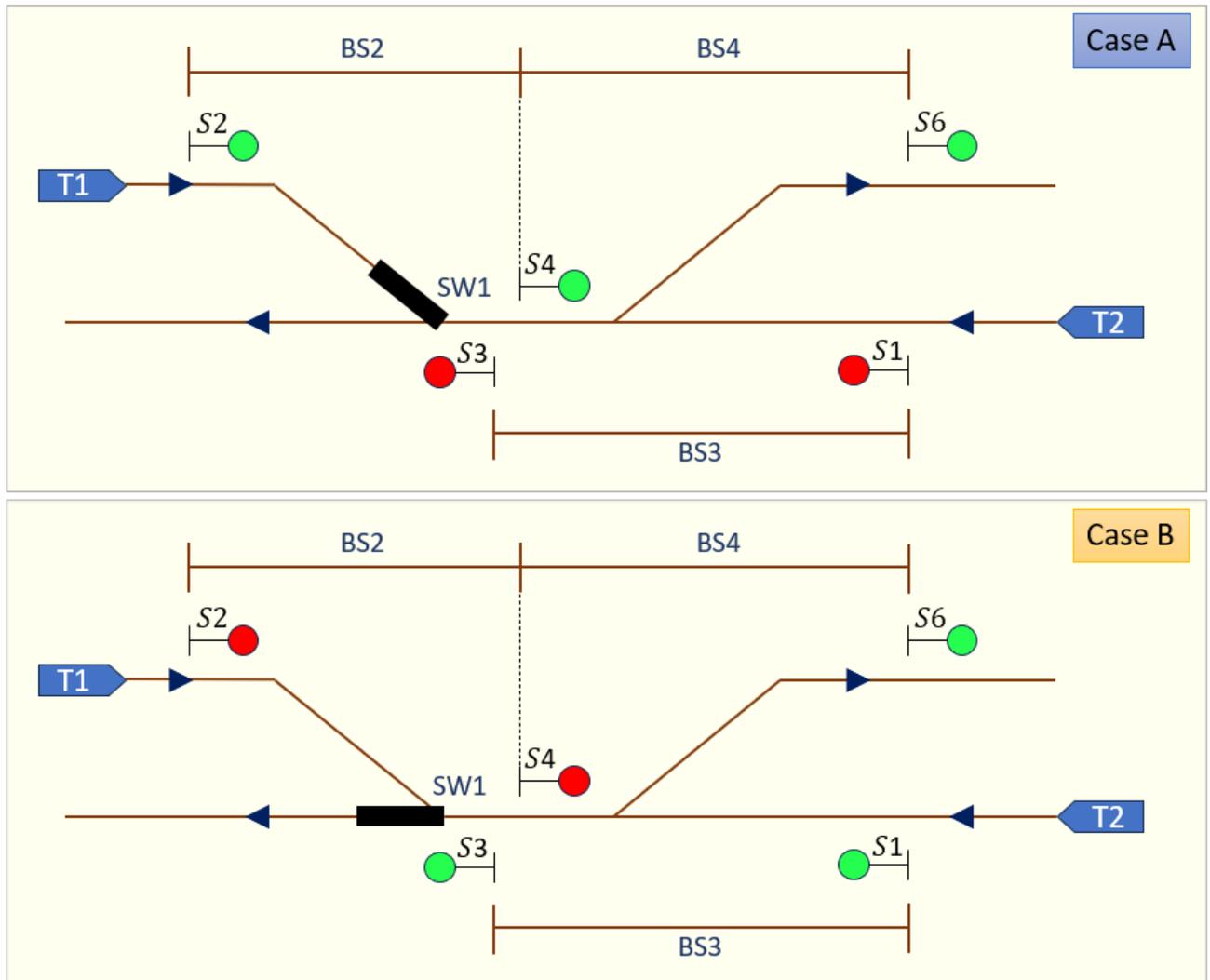
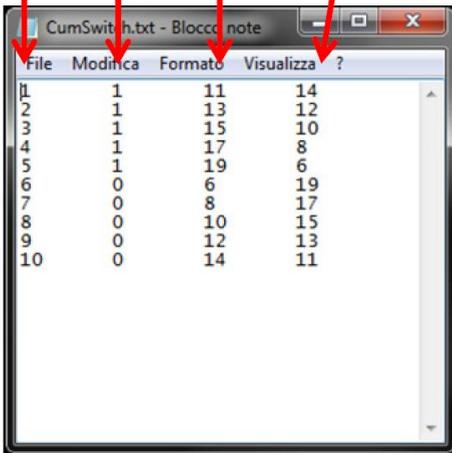


Figure 9. Sample interlocking system with diverging position (Case A) and non-diverging position (Case B)

In this module, the position of movable elements, such as switches, is indicated by a Boolean variable: 1 for the diverging position and 0 for the standard position. Indexes can be specified by the user to select the signals which are set to a proceed aspect and those which are set to a red aspect to prevent the setup of conflicting train routes. For example, in case A in Figure 9, the user can define that when switch SW1 is in the diverging position 1, block section BS2 is accessible, while block section BS3 is blocked to opposing trains. Therefore, this module requires the definition of additional attributes such as 'Switch ID', 'Position of

the switch', 'Index of the block section which shows a proceed aspect when the switch is locked in the position specified by "Position" attribute', and 'Index of the block section which shows a restricted aspect when the switch is locked in the position specified by "Position" attribute' to determine the interlocking components.



Switch ID	Switch position	Block index (proceed)	Block index (blocked)
1	1	11	14
2	1	13	12
3	1	15	10
4	1	17	8
5	1	19	6
6	0	6	19
7	0	8	17
8	0	10	15
9	0	12	13
10	0	14	11

Figure 10. Input database for the definition of the layout of interlocking elements.

Figure 10 shows a database of attributes that define all of the switches on the track. The first row of the database shows that when switch 1 (ID = 1) is locked in the diverging position (Position = 1), block section 11 shows a proceed aspect, while block section 14 is blocked.

When such data are entered within the signalling module, switches are automatically created within the graph model of the infrastructure network, as illustrated in Figure 11.

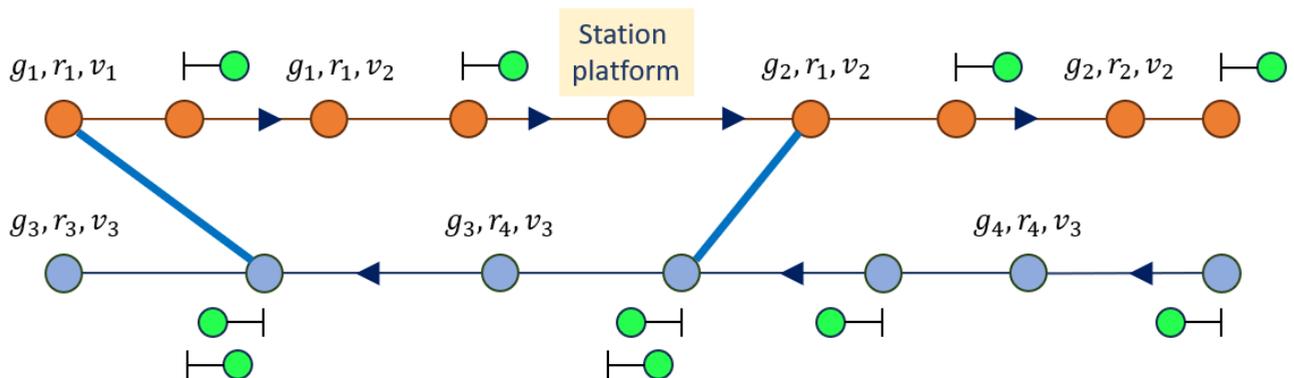


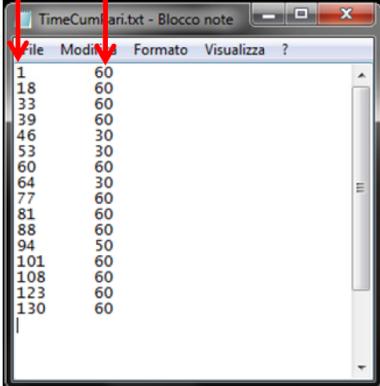
Figure 11. Graph model of the network with signalling and interlocking layout.

2.1.5 Timetable module

The train timetable is a crucial element in managing and organizing train operations within a network. It dictates the routes trains follow, specifies station stops, and outlines departure, arrival, and dwell times. The timetable plays a vital role in determining the chronological sequence of network events. Punctuality and average delay of trains are assessed in relation to the scheduled arrivals and departures defined by the timetable. The timetable module requires input data such as train departure/arrival times or, in the case of regular services like mass rapid transit lines, train headways and dwell times at stations.

The train timetable is represented as a database of all scheduled train operations. Each train object reads its corresponding schedule from this database. For systems with regular traffic, such as mass rapid transit lines, the user must define a train headway (in seconds) as a global variable. Then, the train dwell times at each station must be defined to automatically create a cyclic timetable. To define a cyclic timetable, dwell times must be defined (in seconds) for each station in a database whose fields are 'Station Node ID' and 'Dwell Time'. Figure 12 shows the database containing the aforementioned attributes necessary to define a cyclic timetable. Each record specifies these attributes for a certain line station.

Station Dwell
Node ID time



Station Node ID	Dwell time
1	60
18	60
33	60
39	60
46	30
53	30
60	60
64	30
77	60
81	60
88	60
94	50
101	60
108	60
123	60
130	60

Figure 12. Input database for the definition of the cyclic timetable.

The timetable can be specified for each of the simulated trains by means of text files which report for each of the stopping stations the corresponding planned arrival and departure times as well as the minimum dwell times.

Beside the specification of planned times, the timetable module also allows users to consider stochastic disturbances to train operations, such as random train dwell times at stations or stochastic train entrance delays. Three different probability density functions such as Normal distribution, Negative exponential distribution, Log-normal distribution are available for modelling stochastic dwell times.

The negative exponential distribution is commonly used to model train departure delays, and this module allows users to model train departure delays in the same way. For each type of distribution function, users must specify the average and standard deviation parameters. This can be done by setting the values of input variables in the corresponding global functions in the program code.

The function for defining a normal distribution is *Draw_Normal_Dwell_Time* (double *Average_Dwell*, double *Dwell_StD*). By setting the values of the input variables *Average_Dwell* and *Dwell_StD* in seconds, dwell times for all trains at stations will be distributed as a normal variable with that average and standard deviation.

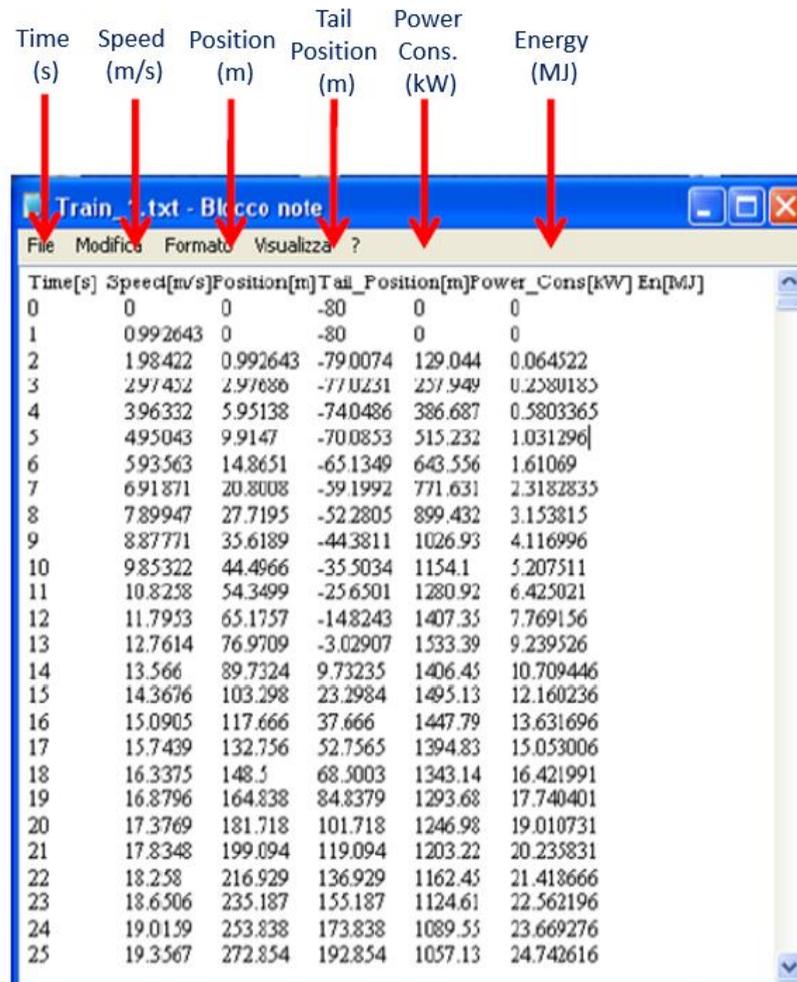
The function for modelling train departure times as negative exponential variables is *Draw_Departure_Delay* (double *Average_Delay*). However, since the standard deviation of a negative exponential distribution is equal to the average, it is only necessary to set the input variable *Average_Delay* in this case.

Users can also set different average dwell times for each station. These values can be taken from the dwell times specified in the timetable database illustrated in Figure 12. Stochastic modelling of train entrance times and dwell times is essential for analysing the stability and robustness of timetables and network infrastructure, as well as for investigating network performances and the effectiveness of certain operation strategies under disturbed conditions.

2.1.6 Outputs of simulation

The microscopic model outputs train diagrams (e.g., speed-distance trajectories, time-distance diagrams) and statistics (e.g., train arrival delays, punctuality at stations). It also outputs diagrams of mechanical power and energy consumed during train runs, and information about the electrical energy supply needed to operate the system under a certain timetable.

When the simulation process stops, a text file is printed for each simulated train, containing all information about the dynamic evolution of that train's state during the whole simulation period. The train outputs are arranged within a database, with fields for simulation time instant, train speed, nose position of the train, tail position of the train, mechanical power consumption, and mechanical energy consumption (Figure 13). Each record in the database contains the values of these fields for a specific time instant in the simulation period.

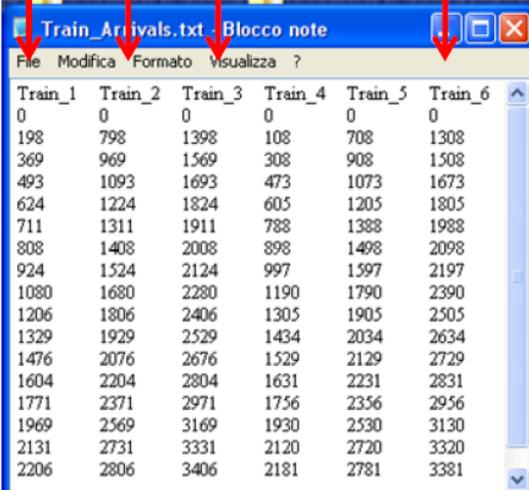


Time (s)	Speed (m/s)	Position (m)	Tail Position (m)	Power Cons. (kW)	Energy (MJ)
0	0	0	-80	0	0
1	0.992643	0	-80	0	0
2	1.98422	0.992643	-79.0074	129.044	0.064522
3	2.97632	2.97632	-77.0231	257.949	0.258018
4	3.96332	5.95138	-74.0486	386.687	0.580336
5	4.95043	9.9147	-70.0853	515.232	1.031296
6	5.93563	14.8651	-65.1349	643.556	1.61069
7	6.91871	20.8008	-59.1992	771.631	2.318283
8	7.89947	27.7195	-52.2805	899.432	3.153815
9	8.87771	35.6189	-44.3811	1026.93	4.116996
10	9.85322	44.4966	-35.5034	1154.1	5.207511
11	10.8258	54.3499	-25.6501	1280.92	6.425021
12	11.7953	65.1757	-14.8243	1407.35	7.769156
13	12.7614	76.9709	-3.02907	1533.39	9.239526
14	13.566	89.7324	9.73235	1406.45	10.709446
15	14.3676	103.298	23.2984	1495.13	12.160236
16	15.0905	117.666	37.666	1447.79	13.631696
17	15.7439	132.756	52.7565	1394.83	15.053006
18	16.3375	148.5	68.5003	1343.14	16.421991
19	16.8796	164.838	84.8379	1293.68	17.740401
20	17.3769	181.718	101.718	1246.98	19.010731
21	17.8348	199.094	119.094	1203.22	20.235831
22	18.258	216.929	136.929	1162.45	21.418666
23	18.6506	235.187	155.187	1124.61	22.562196
24	19.0159	253.838	173.838	1089.55	23.669276
25	19.3567	272.854	192.854	1057.13	24.742616

Figure 13. Text file containing the database of simulation outputs relative to a certain train for a single replication.

In addition to the text files for individual trains, the simulation model also outputs a text file containing train arrival times at each station or key network point. This output is also returned as a database, where each column represents a train and each row shows the train's arrival time at the corresponding station (Figure 14). This information can be used to calculate train statistics on delays and punctuality.

Train 1 Train 2 Train 3 . . . Train j



	Train_1	Train_2	Train_3	Train_4	Train_5	Train_6
Station 1	0	0	0	0	0	0
Station 2	198	798	1398	108	708	1308
.	369	969	1569	308	908	1508
.	493	1093	1693	473	1073	1673
.	624	1224	1824	605	1205	1805
.	711	1311	1911	788	1388	1988
Station i	808	1408	2008	898	1498	2098
	924	1524	2124	997	1597	2197
	1080	1680	2280	1190	1790	2390
	1206	1806	2406	1305	1905	2505
	1329	1929	2529	1434	2034	2634
	1476	2076	2676	1529	2129	2729
	1604	2204	2804	1631	2231	2831
	1771	2371	2971	1756	2356	2956
	1969	2569	3169	1930	2530	3130
	2131	2731	3331	2120	2720	3320
	2206	2806	3406	2181	2781	3381

Figure 14. Text file containing the database of train arrival times at each station for a single replication.

The simulation model also provides functions for calculating the total and average train arrival delays at a certain station. These values can be referred to either a single replication or to the whole simulation experiment. The text files containing this output can be easily imported into software for data analysis (e.g., Excel, Matlab) to generate train diagrams, visualize the effects of train conflicts, and calculate other performance measures or statistics.

2.2 Self-organising operational principles and KPIs

A set of KPI has been implemented in EGTRAIN in line with the main indicators identified in the SORTEDMOBILITY Deliverable D1.1 (“Operational Principles and Key Performance Indicators for Self-Organising Railway Operations”) for assessing self-organising train operations.

The set of KPIs which EGTRAIN can assess are the following:

- *Generalized Travel Time*, which encompasses travel, waiting, and transfer times. This metric provides an overview of the passenger experience, allowing simulation developers and operators to identify areas for improvement in terms of overall journey efficiency.

- *Passenger punctuality*, measured as the percentage of passengers arriving at their destination with less than a certain delay threshold. This KPI allows to assess travel quality from the passenger's perspective considering punctual arrivals as paramount for a positive travel experience.
- *Passenger arrival delays* are also monitored. A delay-related KPI provides insights into the factors affecting passenger travel time, enabling SORTEDMOBILITY to fine-tune self-organizing algorithms to mitigate delays and optimize overall service quality.
- *Train frequency* is a critical KPI in EGTRAIN, influencing the capacity and efficiency of the simulated rail network. By evaluating and adjusting train frequency, the simulator can simulate scenarios with varying levels of service, allowing for the optimization of resources and infrastructure utilization.
- *Total train travel times*, providing a granular understanding of how different factors impact the duration of train journeys. This KPI is useful to monitor the overall efficiency of delivered train operations and helps comparing efficiency of different real-time traffic management strategies.
- Similar to passenger delays, EGTRAIN tracks and evaluates *delays at the train level*. This KPI allows for a detailed analysis of the root causes of delays, helping developers and operators implement strategies to enhance the punctuality and reliability of the simulated train services.
- *Dwell time*, the duration a train spends at a station for boarding and alighting, is measured to evaluate the interaction between passenger movement and train operations at stations.

2.3 Demand models for route choice

In this section we present the details of the route choice model estimated for the simulation platforms in SORTEDMOBILITY. We note that the details of the activity based model are presented in D2.1 Activity-Based Daily Travel Pattern Model. This route choice formulation and estimation is presented in more details in Sfeir et al. (2024), and the latter should be used as reference in any re-used of materials presented here.

2.3.1 Model framework

Public transport route choice models are fundamental components in many transport applications, as they model and predict individuals' route choice behavior and therefore help in assessing and improving public transport network design and performance. A route choice model consists of two main components: 1) choice set generation, which tries to enumerate all possible alternatives between origin and destination pairs; and 2) choice modelling of the chosen alternative from the generated choice set. Within SORTEDMOBILITY, calibration and validation of the route choice model parameters is performed based on data collected for the multimodal public transport network (buses, trains, and metros) in the East Great Belt area of Denmark which includes Zealand, the largest island in Denmark and home to its capital Copenhagen. The Danish Rejsekort (travel card in English) smart card data for travelling public transport in Denmark is used as a reference for setting the parameters of the route choice model also for the other case studies analysed in SORTEDMOBILITY as that was the most complete demand dataset available among all of the considered case studies. Each Rejsekort transaction stores information on the type of transaction (tap-in, transfer, or tap-out), time and location of the transaction, type of the card, and fake card ID (Rejsekort IDs are pseudo-anonymized for privacy concerns). The General Transit Feed Specification (GTFS) data that includes stops, service lines, service line frequencies, and travel time information, in addition to the road network from Open Street Map (OMP) were also used to build the network and compute path attributes such as in-vehicle travel time, waiting time, number of transfers, walking time and path size (to account for path overlaps). Finally, on the supply side we used Banedanmark's and Movia's (bus operator) operating statistics. The Banedanmark dataset provides a comprehensive amount of data on the Greater Copenhagen rail system, spanning all train operations in 2017. It includes important information on Data Exploration, Reliability and Simulation of Travel in the Public Transportation Network train runs, such as planned schedules, train stops, and plan deviations, allowing for a better understanding of the network's structure. The Movia dataset, on the other hand, contains information on bus operations within the region for September 2017. This data includes information on about arrivals and departures for different bus lines, and their stops, providing a detailed overview of bus operations.

2.3.2 Path set generation

A public transport graph was built containing a set of vertices and a set of edges connecting pairs of those vertices. The vertex set consists of bus stops, train and metro stations, and nodes form the road while the edge set connects the vertices with 4 different edge types: bus, train, metro, and walk. Note that an edge connecting a pair of vertices accounts for all common bus lines serving the same pair. The network consists of 11,419 bus stops, 1,425 bus lines, 244 train and metro stations, 46 train and metro lines, 124,368 nodes (from the road network), and 784,603 road segments. To sum up, the graph consists of 136,031 vertices and 3,513,457 edges. In-vehicle travel time was computed as the average scheduled travel time among all service lines on a specific route segment. A constant walking speed of 4 km/h was considered for walking time calculations. Waiting time was calculated based on the overall frequencies of common service lines on a specific route segment as follows:

$$WT_r = \frac{1}{2 \sum_{i=1}^I f_i}$$

Where WT_r is the expected waiting time on route segment r , f_i the frequency of line i , and I the total number of service lines on route segment r .

Next, one day of smart card data (19 September 2017) was selected and all observed stop-to-stop (origin-destination - OD) trips were extracted. The one-day Rejsekort dataset contains, after several cleaning steps, 88,700 unique OD pairs. Then, the conventional choice set was created by generating for each OD pair a set of path alternatives using a combination of four choice set generation algorithms/approaches: k-shortest path, link elimination, labeling, and simulation. In total, paths for 86,128 unique OD pairs were generated for a coverage rate of 98.88%. The high coverage rate was attained by correcting network errors and using a combination of algorithms/approaches. However, due to the large size of the network, building the network and correcting network errors proved to be a costly and time-consuming task that spanned several months. In addition, the choice set generation required two weeks of computation on a Linux system equipped with CPU @ 2.6 GHz and 196GB of RAM. The generated choice set has only 1.21% of OD pairs with one path/alternative and an average of 7.79 alternatives per OD.

2.3.3 Route choice model

Path-size mixed logit models were developed to account for correlation among alternatives due to path overlapping (Hoogendoorn-Lanser et al., 2005) and heterogeneity across passengers.

In public transport networks, path overlapping is not solely limited to overlapping of alternatives along road segments, but it also includes overlapping of boarding stations (Tan, 2016). At each boarding station, passengers have the flexibility to either continue along their current path or change to another one. Therefore, path size factors accounting for roads and boarding stations overlaps will be added to the model.

The utility of choosing alternative i from a generated choice set C_n in choice situation n is expressed as follows:

$$U_{in} = \beta'_X X_{in} + \beta_{PS} PS_{in} + \beta_{PS_{node}} PS_{in}^{node} + \varepsilon_{in}$$

Where X_{in} is a vector of attributes for path i , β_X is a vector of corresponding fixed and random coefficients, PS_{in} and PS_{in}^{node} are path size factors with β_{PS} and $\beta_{PS_{node}}$ their corresponding coefficients, and ε_{in} is a random disturbance term that is independently and identically distributed (*iid*) Extreme Value Type I over decision-makers and alternatives.

PS_{in} is a path-size factor that accounts for correlation due to overlapped road segments along path alternatives and is proportional to travel time as follows:

$$PS_{in} = \sum_{r \in \Gamma_i} \left(\frac{t_r}{T_i} \right) \ln \left(\frac{1}{\sum_{j \in C_n} \delta_{rj}} \right)$$

Where t_r is the travel time on segment r , T_i is the total travel time of all segments on path i , Γ_i is a set containing all segments along path i , and δ_{rj} is a dummy that is equal to 1 if segment r is part of path i and 0 otherwise.

PS_{in}^{node} is a path-size factor that accounts for correlation due to overlapped boarding stops/stations as follows:

$$PS_{in}^{node} = \sum_{s \in S_i} \ln \left(\frac{f_{si}}{\sum_{j \in C_n} \gamma_{sj} f_{sj}} \right)$$

Where S_i is the list of all boarding stops in path i excluding origin (initial boarding stop), f_{si} is the boarding frequency over all shared service lines at boarding stop

s along path i , and γ_{sj} is a dummy that is equal to 1 if s is a boarding stop in path j and 0 otherwise. Under this formulation, a path containing overlapping boarding stations that are served by more frequent service lines will exhibit a large path-size and as such low negative impact on the overall path utility. On the contrary, a path containing overlapping boarding stations that are served with more frequent service lines will have a smaller path-size leading to more negative impact on the overall path utility.

2.3.4 Route choice model parameters

The input parameters of the route choice model presented in this section were obtained by calibrating the route choice model versus the collected Rejsekort data using the software Biogeme. The model was executed to estimate the β parameters and evaluate the impact of various factors on the utility of route choices.

Table 1 presents the performance of the model, while Table 2 shows the estimation of the β parameters.

Table 1. Model results

Parameter	Value
No. of parameters	5
Sample size	165,548
Excluded data	0
Null log likelihood	-303,476.1
Final log likelihood	-121,490.4
Likelihood ratio test (null)	363,971.3
Rho square (null)	0.6
Rho bar square (null)	0.6
Akaike Information Criterion (AIC)	242,990.8
Bayesian Information Criterion (BIC)	243,040.9

Table 2. Parameter estimates

Variable	Value	Rob. Std err	Rob. t-test	Rob. p-value
β_{IVTT}	-0.186	0.003	-73.17	0.000
$\beta_{Nb_transfers}$	-1.398	0.022	-65.08	0.000
β_{PS_j}	0.801	0.022	36.00	0.000
$\beta_{TT_waiting}$	-0.049	0.009	-5.53	0.000
$\beta_{TT_walking}$	-0.506	0.004	-126.70	0.000

In summary, when choosing a route, passengers prefer routes that minimize in-vehicle travel time, the number of transfers, waiting time, and walking time, while

also offering a variety of distinct paths. These factors significantly affect the passenger's decision-making process and ultimately their satisfaction with the transport service. This model effectively captures these key aspects of passenger behaviour, providing useful insights for transport service providers to optimize their services.

Finally, the model was implemented in Python as a stand-alone function and parameterized to allow the modeller to evaluate other set of parameters in simulation if needed. The framework for the interaction with the simulation platforms Opentrack and EGTRAIN were consistent with the proposal described in D4.1, Section 5.2 "EGTRAIN and route choice".

2.4 Self-organising rail traffic management algorithms

The self-organising rail traffic management is realised by means of a software pipeline composed of six modules, as detailed in D3.2 'Algorithms for Self-Organizing Railway Operations'. The pipeline emulates the interaction among the different trains, as autonomous agents, involved in the decentralised decision-making process about the infrastructure utilisation (i.e., routing and timing).

The first module is the neighbourhood identification. Its inputs are the current RTTP (Real-Time Traffic Plan) file and the current TSP file. Here, the RTTP's occupancy times are updated through a time shift utilising the information given in the TSP. Then, the neighbouring trains of every train are identified looking at over-lapping utilisation of the TDSs over any available route within a given time horizon. Finally, a new TSP is produced for any train, where the neighbouring trains are stated.

The second module is the interaction graph generation. Here, given the TSPs generated at the previous step, a graph is produced whose nodes represent the trains of the problem and the edges connect trains belonging to the same neighbourhood.

The third module is the hypothesis generation. Here, a variable number of hypothetical RTTPs is generated for each train, taking into consideration the neighbourhood and a specific objective function. The generation of the hypotheses is implemented as a modified version of RECIFE-MILP, using the information about the trains outside the neighbourhood as constraints.

The fourth module is the hypothesis compatibility. In this module, a compatibility graph is produced whose nodes are the hypothetical RTTPs produced in the previous step and the edges connect only compatible hypotheses produced by trains belonging to the same neighbourhood. Compatibility exists when different solutions can be merged without any infeasibility emergence.

The fifth module is the consensus. Here, the interacting trains' hypotheses are compared, according to their compatibility, resulting on the selection by each train of a single hypothesis to be deployed.

The sixth module is the merge. The module takes the set of hypotheses selected by each train during the consensus and produce a new global RTTP that will be implemented by the traffic control centre. A merge repair submodule intervenes should the global RTTP produced by the merge contain any problem. If consensus is not reached, the original global RTTP is returned.

Lastly, a listener script is implemented which is responsible for interfacing the pipeline with the simulation software.

3 SIMULATION INTERFACE WITH DEMAND MODELS

3.1 Passenger flow simulation module in EGTRAIN

The passenger flow module consists of classes and functions to model passenger trips, journeys, and associated data in a rail transportation system. It serves as a foundation for building a simulation system, allowing for the representation of individual trips, complex journeys, and the dynamic behaviour of passengers within the network. The module integrates with external dependencies to create a more comprehensive simulation environment.

One of the classes defined in module is *trip class* which serves as a blueprint for representing individual trips within the transportation system. Each trip is defined by attributes such as the ID of the Trip, the ID of the departing station, the ID of the arrival station, and the motivation of the trip which indicates the purpose of the trip (e.g., work, leisure). Additional attributes include platform IDs at departure and arrival stations, a description of the train service, planned and actual departure/arrival times, and flags indicating whether the trip has started or completed. The class is initialized through a constructor that sets default values if no explicit values are provided.

Another important class defined in this module is *journey class* which encapsulates a sequence of trips, representing a passenger's entire journey from origin to destination. Attributes include the identification of the journey, the ID of the departing station, and the ID of the arrival station similar to the Trip class, as well as platform IDs, planned and actual departure/arrival times, and variables to track journey progress. Notably, the class contains a list of Trip objects, allowing for representation of complex journeys with multiple transfers. The attributes namely *Planned_Departure_Time_DAS*, *Planned_Arrival_Time_DAS*, *Actual_Departure_Time*, *Actual_Arrival_Time* represent planned and actual times of departure and arrivals of the journey. Actual times are those randomised from the Daily Activity Schedule. *Total_Arrival_Delay* is the overall total arrival delay of the passenger at the final destination of the journey. The total delay is calculated as :

$$TotalArrivalDelay = SimulatedArrivalTime - ActualArrivalTime$$

N_Trips refers to total number of Trips (or also called legs) which compose the Journey. The number of transfers is therefore obtained as

$$Totalnumberoftransfers = N_{Trips} - 1$$

All trips composing the journey is kept in a list. Variables like *Walkingtime* and *Waitingtime* quantify the total walking time in the active journey and the total waiting time spent by the passenger at the platform across the entire journey, respectively.

The passenger class models an individual passenger participating in the transportation system. Attributes include a unique ID of the passenger and the category which categorizes passengers by specifying demographic information such as: elder, minor, adult, or even reduced mobility, normal mobility, or some other attribute relevant to the trip. Other attributes that belong to passenger class are the identifiers for the passenger's current location and the current train service where the passenger is at current time in simulation for a given Trip (*TripID*) in a Journey (*JourneyID*). The class also maintains a list of Journey objects, reflecting the passenger's travel history in a reference period of time. The *CurrentStatus* described whether the passenger is onboard of a train or waiting at a platform after that it entered the simulation.

CurrentStatus can assume value:

- "None" if it is not initialised and the passenger is out of the simulation (Not entered yet or exited)
- "Onboard" if the passenger is onboard of a train
- "OnPlatform" if the passenger is waiting at a station platform to board a train

Current_Train_To_Wait, *Current_Train_Boarded*, *Current_Arrival_Station*, *Current_Arrival_Platform* are attributes which refer to the name of the train that the passenger is waiting for, name of the train that the passenger is boarding, name of the arrival station and the specific arrival platform, respectively. Similarly *Current_WaitingStationPlatformID*, and *Current_WaitingStationID* are the ID of the station platform and the station ID where the passenger is waiting at. Also, there is a Boolean variable namely, *IsIntheNetwork*, to check whether the passenger is within or without the rail network in the simulated time window. This variable is true when the passenger enters the simulation (i.e. simulation time \geq Journey start time) and when the passenger has exited the simulation as it reached its Journey destination. The UML representation of trip, journey and passenger classes can be found in Figure 15.

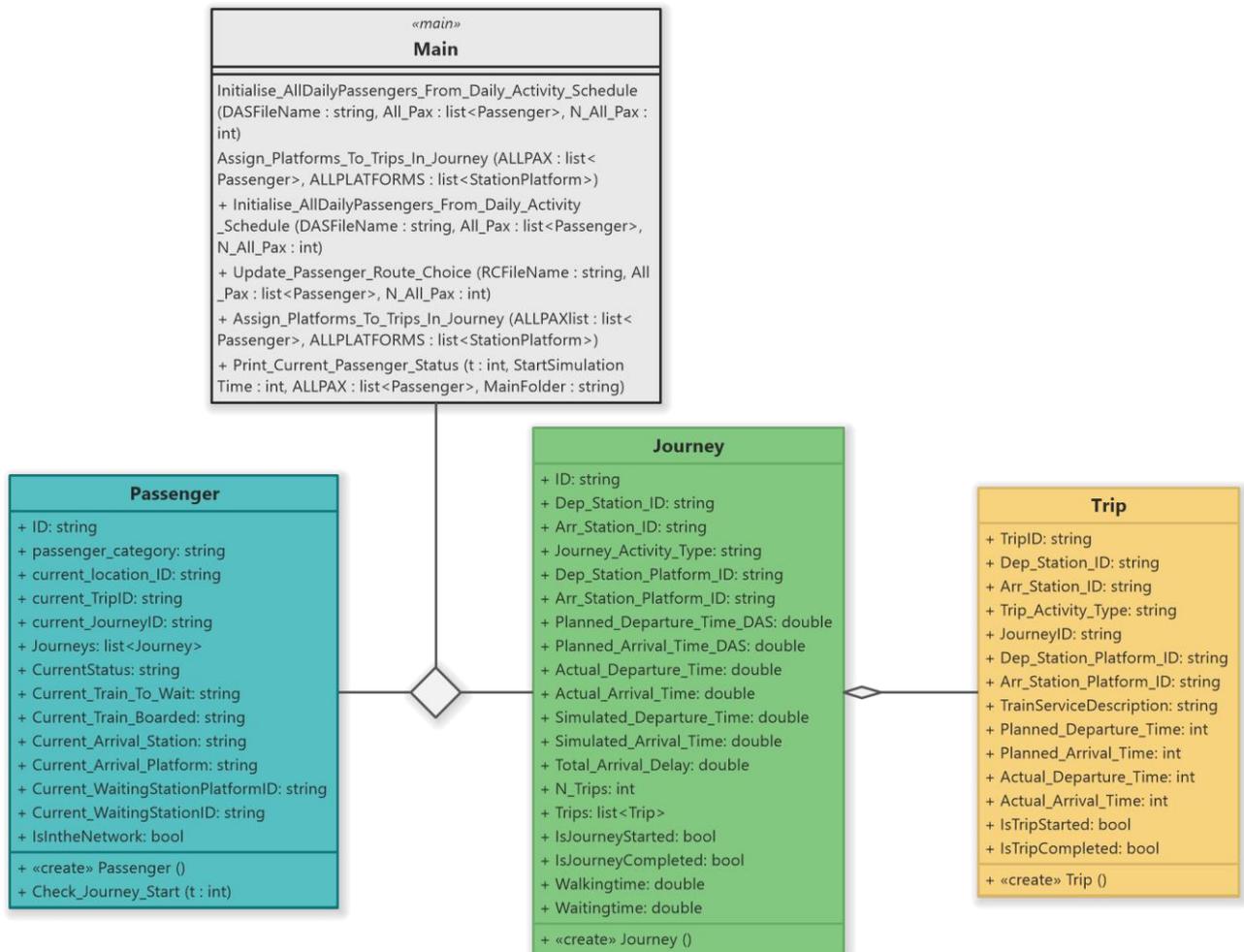


Figure 15. UML Class representation of Passenger module.

There are a global list and a global variable in the passenger module. The *AllDailyPassengers* variable is a list containing all passengers in the network throughout the day while *N_AllDailyPassengers* variable stores the overall number of passengers using the rail service across the day.

In order to generate passengers in the network based on Daily Activity Schedule (DAS) and Route Choice, four functions are defined. These functions initialize the passenger list based on a daily activity schedule, modify passenger route choices, and assign platforms to trips within a journey. Besides that, a function is called to print the current status of passengers at a specific simulation time.

3.2 Interaction with daily activity schedule and route choice

The flow chart given in Figure 16 shows the relationship of daily activity schedule and route choice files with the passenger module newly added to EGTRAIN. The DAS file produced by SIMMobility is taken as input by the passenger module. The board/alight status of the passengers is updated by taking into account the traffic status information from the EGTRAIN rail traffic simulation core, the passenger information received from the DAS file and the trains on the platform. The traffic situation based on current passenger data is shared with a variant of RECIFE-MILP (see D3.1 'Algorithms for Self-Organizing Railway Operations') and a new RTTP is generated if necessary. Current passenger data is eventually transferred to the Route choice model to ensure consistency between the actual passenger state and the set of available route alternatives based on the updated RTTP.

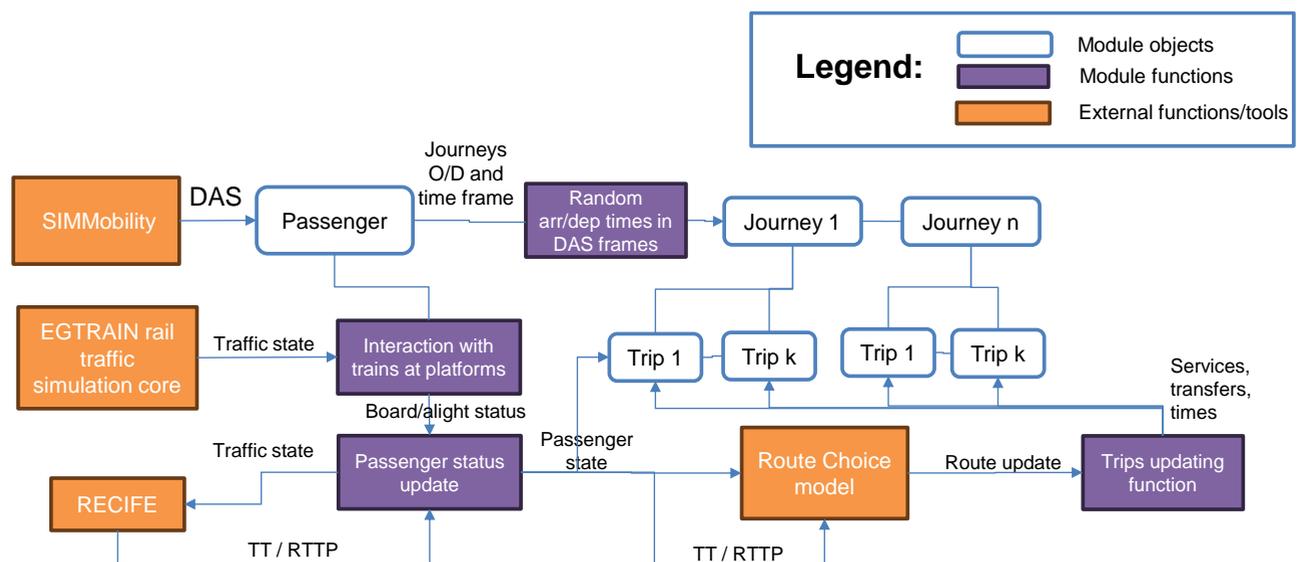


Figure 16. The interaction with daily activity schedule and route choice

3.3 Passenger-dependent train dwell time estimation

Dwell time in transportation typically refers to the amount of time a vehicle (such as a bus, train, or tram) spends at a station or stop for boarding and alighting passengers. The factors affecting dwell time can include the number of passengers, the efficiency of the boarding/alighting process, any delays, and other operational considerations.

Fernandez et al. (2007) developed a microscopic simulation model (PASSION) to study the interaction between buses and passengers at bus stops. This model

was developed to overcome the limitation of the IRENE model presented by Gibson et al. (1989). The calculation of the dwell time of a public transport vehicle at a certain stop, according to Gibson et al. (1989), is as follows:

$$ST = \beta_0 + \beta_1 \delta_1 + \max_j$$

In which, PA_{ij} refers to the set of passengers alighting from vehicle i through door j and PB_{ij} representing the number of set of passengers boarding vehicle i through door j . As for the dummy variables delta, they will take the value of 1 in the following cases:

- Congestion at platform $\delta_1 = 1$ if the degree of congestion at the stop platform (ratio between the number of passengers waiting and the maximum number of passengers that the quay can accommodate) is greater than 0.65.
- Congestion at the boarding process $\delta_2 = 1$ if more than 15 passengers board the vehicle at stop j
- Congestion on board the vehicle $\delta_3 = 1$ if the degree of filling on board the vehicle (ratio of the number of passengers in the vehicle to the maximum number of passengers that the vehicle can hold) is greater than 0.7.

Otherwise, the value of the dummy variables will be 0.

Regarding the parameters of the dwell time function (β_i), it is necessary to implement a data collection and calibration process to provide the accurate values for the parameters. For this implementation, the values of the parameters are taken from a previous calibration done at the metro of Santiago de Chile. The default values of the parameters set in EGTRAIN are shown in Table 3. However a specific calibration phase is performed against collected dwell times data to set the values of those parameters which best describe observed dwell time-passenger volume dependency for a given case study.

Table 3. Default parameters of Gibson dwell time model

β_0	β_1	β_2	β_3	β_4	β_5	β_6	β_7
5.631	0.000	0.815	0.564	0.016	2.012	0.086	0.562

Stopping times of trains are changed based on the Gibson dwell time function at the platforms as they arrive. The stopping time is not changed if an additional passenger arrives last minute as it is assumed that the platform will be less

crowded and the passenger can enter with no additional dwell time required with respect to that necessary to alight/board all passengers when the train approached the platform.

4 SIMULATION INTERFACE WITH ALGORITHMS FOR SELF-ORGANIZING TRAFFIC MANAGEMENT

In D4.1, The comprehensive self-organizing simulation platform was drafted by identifying distinct modules such as the control architecture and the transportation (simulation) system and their corresponding interactions. Furthermore, it is noted that in order to facilitate the exchange of input and output data among these diverse modules, several interfaces have been established (see Figure 17).

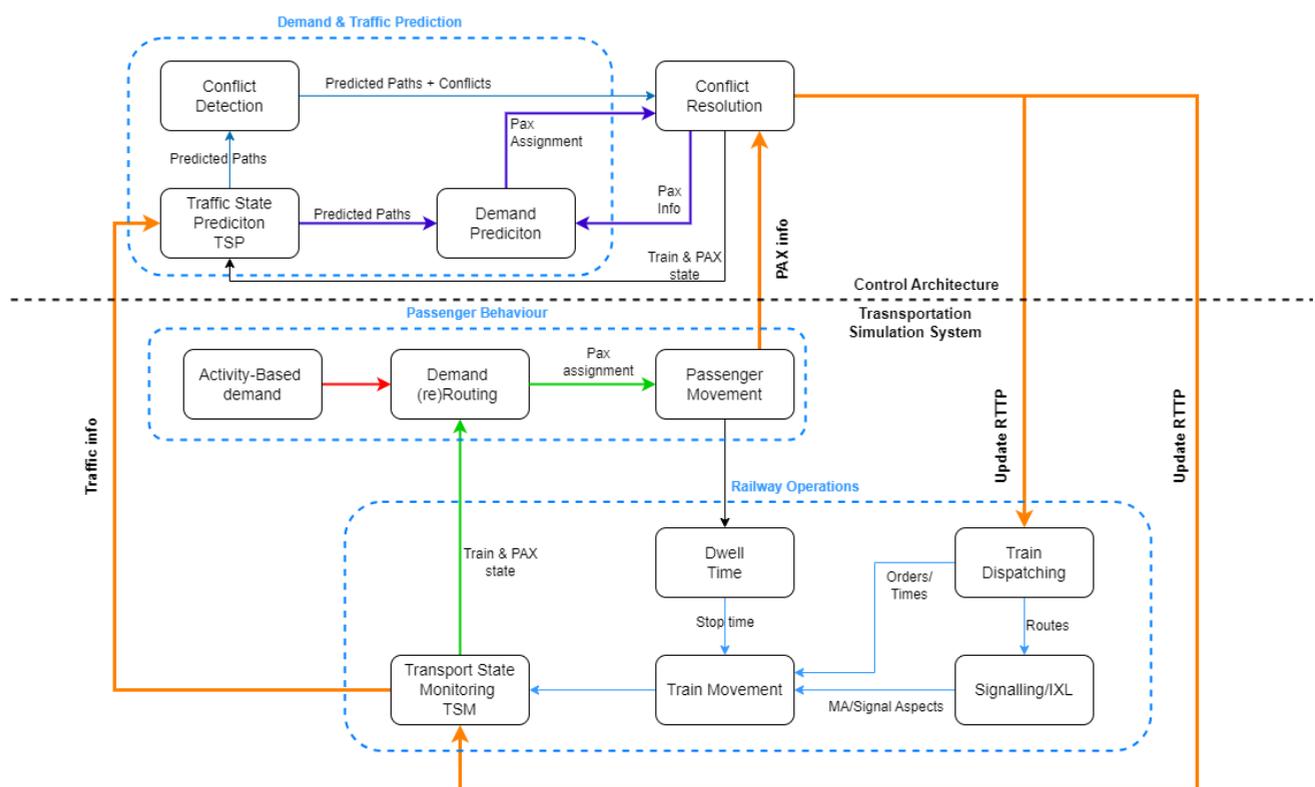


Figure 17. Software interfaces

4.1 Transport State Monitoring module in EGTRAIN

As described in the deliverable D4.1, the simulation platform communicates the state both passengers and trains in the network to the Demand and Traffic Prediction modules using the template described in Deliverable 4.1. EGTRAIN sends the "Traffic State Monitoring" (TSM) file at each Scheduling Interval, namely the duration of the simulation cycle between two calls of Demand and Prediction module.

The TSM represents a snapshot of the current state of the railway. It is given with the current time and the list of trains involved in the problem. For the trains that are currently utilizing the railway, the last occupied TDS are indicated along with the time at which the occupation was detected. For the trains not yet departed, the expected entrance time is provided. Besides the traffic state, the demand state is included in the TSM. To do so, information is added about the current status of passengers in terms of their position in the network, whether on board of a train or waiting at a station platform as well as their current destination station. A web-based AMQP data architecture has been developed to allow the exchange of TSM and RTTP with the self-organising traffic management algorithms based on Zero-MQ.

```

<trafficState currentTime="2022-10-21 01:02:00.0 CET">
  <trainStateInArea>
    <trainID trainNumber="T1"/>
    <trainPosition currentTrackVacancyDetectionSection="TDS013"
      lastOccupationTime="2022-10-21 01:01:30.0 CET"/>
  </trainStateInArea>
  <trainStateInArea>
    <trainID trainNumber="T2"/>
    <trainPosition currentTrackVacancyDetectionSection="TDS156"
      lastOccupationTime="2022-10-21 01:01:20.0 CET"/>
  </trainStateInArea>
  <trainStateOutArea>
    <trainID trainNumber="T3" expectedEntranceTime="2022-10-21
      01:07:30.0 CET"/>
  </trainStateOutArea>

```

Figure 18. TSM.xml file without passenger information

The sent TSM.xml file has been extended to include also the passenger information as shown in Figure 19. In the extended format, each passenger entering or exiting the network is listed with his unique id as well as the time that he entered or exited the network.

```

<trafficState currentTime="2023-05-01 01:02:00.000 CEST">
  <passengerData>
    <stoppingGroup stoGro_Id="BES">
      <TapIn>
        <passenger pax_Id="1000" time="07:10:01" />
        <passenger pax_Id="1001" time="07:10:02"/>
        <passenger pax_Id="1002" time="07:10:03"/>
      </TapIn>
      <TapOut>
        <passenger pax_Id="2000" time="07:30:04"/>
        <passenger pax_Id="2001" time="07:30:05"/>
        <passenger pax_Id="2002" time="07:30:06"/>
      </TapOut>
    </stoppingGroup>
    <stoppingGroup stoGro_Id="GA">
      <TapIn>
        <passenger pax_Id="3000" time="07:20:01" />
        <passenger pax_Id="3001" time="07:20:02"/>
        <passenger pax_Id="3002" time="07:20:03"/>
      </TapIn>
      <TapOut>
        <passenger pax_Id="4000" time="07:20:04"/>
        <passenger pax_Id="4001" time="07:20:05"/>
        <passenger pax_Id="4002" time="07:20:06"/>
      </TapOut>
    </stoppingGroup>
    <stoppingGroup stoGro_Id="PA">
      <TapIn>
        <passenger pax_Id="2000" time="07:10:01" />
        <passenger pax_Id="2001" time="07:10:02"/>
        <passenger pax_Id="2002" time="07:10:03"/>
      </TapIn>
      <TapOut>
        <passenger pax_Id="1000" time="07:30:04"/>
        <passenger pax_Id="1001" time="07:30:05"/>
        <passenger pax_Id="1002" time="07:30:06"/>
      </TapOut>
    </stoppingGroup>
  </passengerData>
  <trainStateInArea>
    <trainID trainNumber="T00009919B"/>
    <trainPosition trackID="SEC_3" travelDirection="-1" posOnTrack="170.0"
      currentTrackVacancyDetectionSection="TDS000013"
      previousTrackVacancyDetectionSection="TDS00011" lastOccupationTime="01:01:30 CEST"/>
    <speed>0.0</speed>
  </trainStateInArea>
  <trainStateInArea>
    <trainID trainNumber="T00009922B"/>
    <trainPosition trackID="SEC_58" travelDirection="1" posOnTrack="50.0"
      currentTrackVacancyDetectionSection="TDS000156"
      previousTrackVacancyDetectionSection="TDS000152" lastOccupationTime="01:01:20 CEST"/>
    <speed>27.0</speed>
  </trainStateInArea>
  <trainStateOutArea>
    <trainID trainNumber="T00009920B" expectedEntranceTime="01:07:30 CEST"/>
  </trainStateOutArea>
</trafficState>

```

Figure 19. TSM.xml file with passenger information

4.2 Traffic management module in EGTRAIN

Upon receiving information on current traffic and passenger status the traffic management algorithms for conflict detection and resolution return a Real-time Traffic Plan. Such a RTTP is sent to the Traffic Management module of the traffic simulation platform such that trains, the signalling and interlocking systems are instructed to implement the updated train orders, times and routes provided by the RTTP. The RTTP file is described in the form of a standard xml as show in Figure 20. The duration of the RTTP is determined by the so-called Prediction Horizon (i.e. the time ahead in the future in which traffic conflicts are predicted and solved) which is set as an input parameter to the algorithms for traffic conflict detection and resolution.

```

<rTTP>
  <rTTPTrainView>
    <rTTPForSingleTrain journeyId="DC1D1D__CL" trainID="T004005 ">
      <tDSectionOccupation tDSectionID="TDS120" trainID="
T004005" occupationStart="3720" routeId="R95"
trainSequenceID="0"/>
      <tDSectionOccupation tDSectionID="TDS112" trainID="
T004005" occupationStart="4117" routeId="R81"
trainSequenceID="1"/>
      <tDSectionOccupation tDSectionID="TDS111" trainID="
T004005" occupationStart="4139" routeId="R81"
trainSequenceID="2"/>
    </rTTPForSingleTrain>
  </rTTPTrainView>
  <rTTPInfrastructureView>
    <rTTPForSingleTDSection tDSectionId="TDS081">
      <tDSectionOccupation tDSectionID="TDS081" trainID="T1"
occupationStart="3727" routeId="R50" trackSequenceID="0"/>
      <tDSectionOccupation tDSectionID="TDS081" trainID="T2"
occupationStart="5989"routeId="R52" trackSequenceID="1"/>
      <tDSectionOccupation tDSectionID="TDS081" trainID="T8"
occupationStart="7078" routeId="R52" trackSequenceID="2"/>
    </rTTPForSingleTDSection>
  </rTTPInfrastructureView>
</rTTP>

```

Figure 20. RTTP.xml file

The RTTP contains the traffic plan in two different forms, the “infrastructure view” and the “train view”. The former lists for each Track Detection Section (TDS)

which train and at what time it starts occupying them the Prediction Horizon whereas the latter lists for each train the starting occupation time of each Track Detection Section they cross over their routes. Both forms include the same information.

In the following the micro-simulation platform updates train times, orders and routes according to the plan provided by the RTTP. The outcoming new traffic state is then again sent to the Demand and traffic prediction module and the cycle of information iterates again.

4.3 Interaction with self-organising traffic management algorithms

As already introduced in D4.1, the interaction between the simulation platform and the traffic management module is executed by exchanging information about the traffic state and plans. The simulation software utilizes two different types of information, static and dynamic, which are shared between the simulation platform and the control architecture.

The static information includes the railway infrastructure, the planned timetable and the rolling-stock characteristics, which are shared in the RECIFE data format, i.e., a xml representation compatible with railML. The detailed description is available at http://recife.univ-eiffel.fr/sharedData/data_format_documentation/.

The dynamic information is represented by the RTTP and the TSM files.

As mentioned above, in order to let the pipeline interact with the simulation platform, a listener module is implemented. It waits for a TSM to be ready, as produced by the simulation platform, and communicated to the pipeline (e.g., the TSM is copied in a given folder). As soon as the TSM is received, the listener executes the pipeline. When the pipeline terminates, the listener renames the used TSM and RTTP keeping track of the utilisation order, then communicates the new global RTTP to the simulation platform for use in the next cycle and waits again for a new TSP file.

4.4 Traffic management communication interface

A web-based AMQP data architecture has been developed to allow the exchange of TSM and RTTP with the self-organising traffic management algorithms based on ZeroMQ. ZeroMQ, also known as ØMQ, is a high-performance messaging library renowned for its simplicity, flexibility, and efficiency (ZeroMQ, 2023). It

facilitates scalable and distributed messaging patterns and supports diverse communication models such as request-reply, publish-subscribe, push-pull, and two-way communication. The library accommodates multiple transport protocols, including TCP, IPC (Inter-Process Communication), and in-process communication, allowing developers to choose the most suitable transport mechanism for their specific requirements.

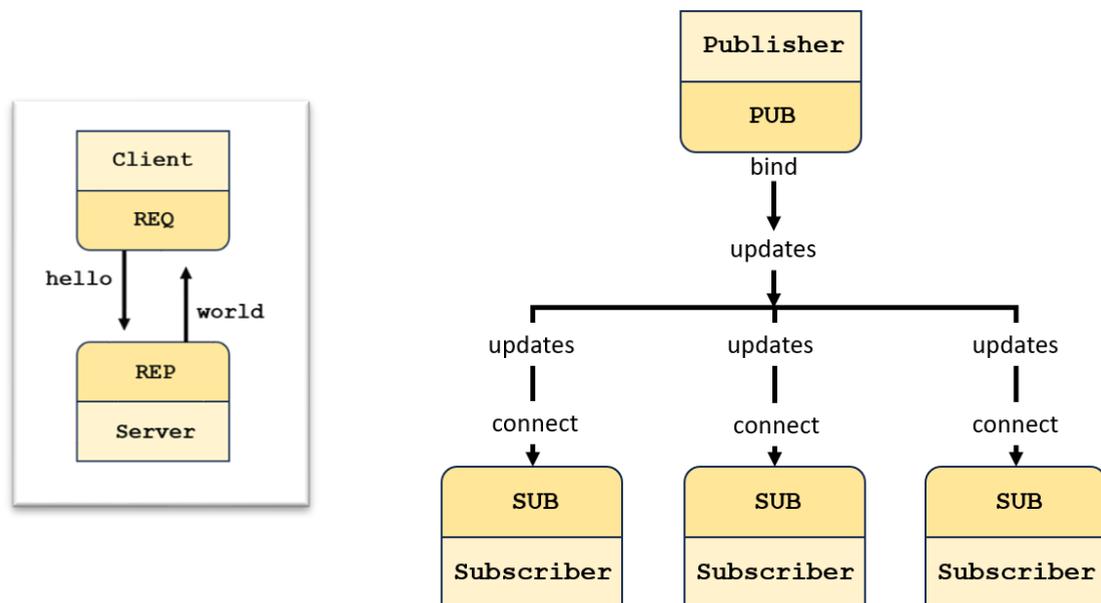


Figure 21. Flow diagram of ZeroMQ.

The general framework of ZeroMQ in terms of subscriber and publisher relation is given in Figure 21. ZeroMQ employs various socket types to define communication patterns, such as request-reply sockets (ZMQ_REQ and ZMQ_REP) and publish-subscribe sockets (ZMQ_PUB and ZMQ_SUB). It excels in asynchronous operation, enabling efficient communication between nodes without synchronous blocking, contributing to its high performance. Notably, ZeroMQ adopts a brokerless design, promoting a peer-to-peer architecture without a central server or message broker. This decentralized architecture simplifies system design and enhances scalability, making it a popular choice for building distributed systems that can scale horizontally.

5 CONCLUSIONS

The research contained in the deliverable has the main objective of specifying and detailing the interactions among the different software modules and algorithms of the SORTEDMOBILITY simulation platform to assess self-organising rail traffic operations. For each of the modules and algorithms described in Deliverable D4.1 an overview of the main functionalities and input/output data is provided. An accurate description is then given of the microscopic simulation tool EGTRAIN which is used as an emulator of a real traffic environment in the assessment architecture. Core simulation functions as well as input and output data are reported together with novel modules specifically developed for enabling the interaction with the travel demand and self-organising traffic management algorithms developed in SORTEDMOBILITY. Specifically, a functional description including mathematical formulation and data flow is provided for the newly-built modules for passenger flow simulation and Traffic management as well as additional KPIs as defined in SORTEDMOBILITY deliverable D1.1. The deliverable also reports details about the web-based AMQP data interfaces based on ZeroMQ and the standardised data format which enable flexibility of the architecture allowing interchangeability of the underlying rail traffic simulation environment.

6 REFERENCES

- E. Quaglietta. A Microscopic Simulation Model For Supporting The Design Of Railway Systems: Development And Applications. University of Naples Federico II, Italy, 2011.
- E. Quaglietta, P. Pellegrini, R.M.P. Goverde, T. Albrecht, B. Jaekel, G. Mar-lière, J. Rodriguez, T. Dollevoet, B. Ambrogio, D. Carcasole, M. Giaroli, and G. Nicholson. The ON-TIME real-time railway traffic management framework: A proof-of-concept using a scalable standardised data communication architecture. *Transportation Research Part C: Emerging Technologies*, 63:23–50, 2016.
- G. Sfeir, F. Rodrigues, R. Seshadri, and C. Lima Azevedo. Identifying Choice Sets for Public Transport Route Choice Models using Smart-Card data: Generated vs. Empirical Sets. Submitted to the 17th International Conference on Travel Behaviour Research July 14 - 18, 2024 – Vienna, Austria, 2024.
- J. Gibson, I. Baeza, & L. Willumsen. Bus-stops, congestion and congested bus-stops. *Traffic Engineering & Control*, 30(6). <https://trid.trb.org/view/296139>, 1989.
- R. Tan, M. Adnan, D. Lee, & M. Ben-Akiva. New path size formulation in path size logit for route choice modelling in public transport networks. *Transportation Research Record: Journal of the Transportation Research Board*, 11–18, 2015.
- R. Fernandez, V. Burgos, & C. Cortés. Results of the microscopic modelling of traffic interactions at stops, junctions and roads for the design of bus rapid transit facilities, 2007.
- S. Hoogendoorn-Lanser, R. Van Ness, & P. H. L. Bovy. Path Size and Overlap in Multi-modal Transport Networks. *Transportation and Traffic Theory. Flow, Dynamics and Human Interaction*. 16th International Symposium on Transportation and Traffic Theory, 2005.
- W.J. Davis jr., The Tractive Resistance of Electric Locomotives and Cars, *General Electric Review*, vol. 29, 1926.
- ZeroMQ – the website: <https://zeromq.org>, accessed in December 20, 2023.