Politecnico di Torino

Master of Science in Civil Engineering

Simulation of intermodal terminals and transhipment techniques in road-rail combined transport

Tutors:

Prof. Ing. Bruno DALLA CHIARA

Ing. Nicola COVIELLO

Candidate:

Guillermo Ernesto GIRARDI

September 2018

Index

Figure Index

1. Introduction

With the large amount of computer tools available today, an evolution in the way we visualize the different engineering activities is inevitable. In this thesis, I will present an application of a very broad software such as MatLab® to a concrete engineering case, the simulation of intermodal terminals.

The principal scope is to create a useful method for the design and simulation of Inland type intermodal terminals, by means of the previously mentioned software, where valuable data can be obtained through the results of this simulation.

1.1. Framework

An intermodal terminal is the infrastructure where the modal shift between different modes of transport takes place using a form of standardised container, the Intermodal Transport Unit (ITU), allowing for a more rapid and efficient transhipment of goods and materials. The ITU can be of the container type, swap bodies and semi-trailers. The first are parallelepiped-shaped cargo containers that protect merchandise from the weather and are manufactured in accordance with ISO (International Organization for Standardization) standards, specifically ISO-668;2 for this reason, they are also known as ISO containers, the most common being those 20 feet long and 40 feet long. The swap bodies are apparently similar to the containers, but with greater ease of unloading from the truck to the railroad or ro-ro transhipment vessel, due to the use of four removable bases incorporated in the frame, which allow them to remain alone under the loading and unloading harrows in the warehouses. In contrast to containers, they are not stackable. Finally, semi-trailers are non-motorized transport vehicles that unload part of their weight onto their own axles and are made to be easily coupled and uncoupled from the truck that pulls them. There are diverse types of intermodal transport, this thesis focuses on combined land transport (road and rail) which is internationally defined as "inland terminal".

Analysing the trends, we can see a growth in recent years of combined transport in Europe, reaching in 2017 to transport 3.2 million shipments by UIRR member countries, values never seen before. There has also been a very large growth, of the order of 38%, in Europe's transport to and from Asia [1]. Taking advantage of such a favourable context, new and better tools should be developed and why not increase accessibility to them. The last is the objective of this thesis.

I will deal with a simple case, but without losing the generality to have the possibility of expanding it and to be able to simulate terminals of different size and layout.

This thesis is intended for both the operational part of a terminal, where one can know the impact of the different variables such as the breakage of one of the cranes, or of the locomotives, hourly delays in trains, number of broken cars, among others, as well as for professionals who want to design new terminals, or expand existing ones, to have a tool to help define the type of plant (lay-out), number of cranes, potential, etc. Another useful point of view could be the application of simulation for educational purposes.

The first part will deal with the state of the art on the subject, then I will deal with the assumption and simplifications taken into account for the design to proceed to explain in detail the creation of the simulation model implemented in MATLAB, concluding with the simulation of different feasible scenarios and their analysis.

1.2. Aims and Specific Purposes

The main objective is to create and code an efficient, useful and easy-to-apply method for the design and simulation of Inland type intermodal terminals, using a versatile, all-purposes software that is found in all study areas and is constantly updated as Matlab ®. It should be borne in mind that Matlab, or any of its packages, have not been specifically designed for the simulation of intermodal terminals, in reality they are generic software that require a great deal of study and knowledge to carry out complex simulations, as in our case, since not only are queue-type events generated in a station, which are easily simulated, but also more complex events such as the exchange of ITUs and the compatible movement of trains and locomotives within the station that require creativity in the combination of the multiple tools included in the software.

In addition, this software has the possibility of obtaining multiple results as well as the time of use of the different available resources, such as cranes and locomotives, and the use of each sub-system that allow us to obtain quantitative values in order to make an objective comparison between the different modifications that can be carried out in an intermodal station. It is also possible to know if service queues are generated in each of the main parts of the station, their average length, the average values of related waiting time, among other parameters that allow us to carry out a detailed study of the station itself, generating abundant information to facilitate decision making.

Specific objectives that are proposed and are within the above are:

- Possibility of improving the understanding of intermodal stations through their simulation.
- Highlight the variables within an intermodal station that are important for its management. Observe the influence of the change in these variables on the day-to-day running of an intermodal terminal.
- Get the most information in the simplest and most intuitive way through the results of the analysis of the whole operation.
- Generate an efficient method that can be used in accordance with the complex development of these operations.
- In general, make a simulation model that is useful for as many people interested in the subject as possible.

2. State of the Art

Knowledge of how to design intermodal terminals, how to operate them and how to produce intermodal transport is not only tacit knowledge within intermodal operators, designers and engineers, but has also been transferred and developed by universities and engineering consultancies. The latter have developed models and decision support systems for the assessment of terminal networks and intermodal terminals. Among the support systems developed is the use of simulation software for both the intermodal networks and the terminal itself. The amount of material in this field is very large, but there are also many models and support systems developed in-house.

We can cite the MINT project - Model and decision support systems for intermodal terminal networks - developed for the European countries of Austria, Germany, Norway, Sweden and Switzerland in 2011. This project consisted of obtaining detailed reports about intermodal transport at the time. Among his publications, I will highlight the third one, which deals with the modelling and simulation of intermodal terminal networks.[2] In this publication numerous software used in research are named:

- EvaRail: Model for Economic evaluation of rail freight services.

Activity-based rail transport costing model developed at the Royal Institute of Technology in Stockholm (KTH). The aim of EvaRail is to describe in detail the supply and demand of rail freight transport. The model provides cost data both at the system level and at the flow/embarkation level (micro and macro level). At the same time, it is general in the sense that it can represent (almost) all production systems and combinations of production systems in rail freight, both those currently in use and those to be used in the future. In this way, the model can describe railway development opportunities in a more dynamic perspective, considering changes in railway supply because of new production methods, improved technical solutions and operational performance.

The scope of an economic-business model is affected by the organisational structure of the sector. In the rail industry - as in other industries - this structure is changing over time, and there are also differences between different nations and railway operators. EvaRail is designed in such a way that to some extent it can be adapted to different organisational models. [3]

- SimConT: Simulation of Inland Container Terminals

Model of the scope of the detailed analysis of terminals. "SimConT is a decision-support tool, based on modern simulation techniques, for efficient resource-planning and effective capacity utilization, which helps in planning of operations of inland container terminals (ICT). In the current version the tool takes into consideration all operations of bimodal terminals (road and railway). Further developments will include the extension to trimodal traffic. With the help of the SimConT tool, planned changes in the complex systems of inland terminals can be tested on their strategic, tactical and operational impacts".[4]

SimConT is not restricted to national standards, conventions or other barriers, since it is a simulation tool tailored to the characteristics of inland terminals, which need more or less the same infrastructure all over the world. SimConT is tailored to the special needs of inland container terminals (mix of loading units, arrival patterns of trains, focus on train-train transhipment). A qualified implementation in open sea terminals is possible. The transfer to other actors, having the same interest in intermodal infrastructure is possible.

- TermCost: calculates terminal costs for infrastructure and operation.

The TermCost model has been developed as a standalone model with the aim to be able to calculate the transshipment costs for a certain time period based on a given input and resource configuration, given a maximum utilisation.

Used as a standalone model or in combination with SimConT, different terminal alternatives can be evaluated. By changing the objects and attributes, different alternative scenarios might be evaluated relatively to a base scenario.

The results of such an analysis might be, e.g. (1) the total costs for a certain time period, (2) the costs per handled TEU for a given time period and (3) costs per input attribute type for a given time period. The input attribute types in the basic configuration are: (1) investment costs, (2) maintenance costs and (3) operational costs.[5]

- SimNet: Terminal network simulation model.

The SimNet model is conceived as a multi agent simulation tool which intends to compare the performance of intermodal networks, given different network settings (physical network topology or functional network structure) and different operational concepts (train concepts, truck arrivals). Further, the model aims to coordinate the flow of load units in the network in case of network element overload. The network performance can be evaluated in terms of the throughput time of the load units, overall transportation costs (given cost factors for the usage of the different network nodes and links) or utilisation measures of the individual network elements.[6]

Other software used in terminal simulation is the AutoMod, where industrial plants of any level of complexity and detail can be accurately simulated with both manual and automatic operations.[7] Like Matlab, AutoMod is not specific software for the design of intermodal stations, its character is general and much effort must be made to encode efficiently to obtain acceptable results by using the predefined blocks available in the program. AutoMod helps increase the effectiveness and efficiency of complex plant and system designs with its 3-D Virtual Reality Graphics, Interactive Modelling, Spreadsheet Interface, and Predefined Handling Systems features. It is possible to design the conveyors and define their characteristics: speed, acceleration and deceleration, type of accumulation, stopping and moving space, pitch, photocells, motors, etc.[8]

Another software very important and used is The Vision Traffic Suite, which is a software-based transportation planning, traffic engineering and traffic simulation software package, specifically PTV Vissim: Container Terminal and Rail Simulation, is used in intermodal terminals.[9] VISSIM offers a COM programming interface allowing users to integrate VISSIM in their own applications using languages like Visual Basic, Python or C++. The COM interface provides access to the network topology, signal control, path flows, vehicle behaviour and evaluation data. Typical applications include automation of work flow processes, modification of simulation parameters during run time, and customised display options. Important is that COM allows full flexibility and thus empowers the user to use his or her own creativity fully [10].

At the University of Zilina, Slovakia, Simcon's Byron software was used to create a simulation model of the transport (logistics) centre [11]. This model allows the different processes to be modelled and simulated in the same configurable application. The objective of the simulation model is a logistics centre, which can contain any combination of modules: marshalling yard, warehouse and intermodal terminal.

The marshalling yard is an important centre in the process of rail transport. The classification station simulation model allows users to choose from several standard configurations of each track group and simulate several variants of technological processes. The warehouse is the object or area specifically designed and designed for the storage of goods, which is equipped with storage technology and equipment for the reception, storage, handling, repair and distribution of goods. And the intermodal terminal provides a basic preview of the infrastructure and processes at the terminal.

The work relates, through the Byron software, the three aspects mentioned above, concluding that the planning and optimisation of infrastructure, scheduling and the sources of management practices in transport centres cannot be carried out without a direct and objective assessment of the consequences of decisions. In addition, it clarifies that computer simulation is the primary method used to find answers to specific, customized questions and problems.

In the 9th International Conference on Ambient Systems, Networks and Technologies (ANT 2018) was introduced from Publish-subscribe for intermodal terminals [12]. Publish-subscribe is a communication pattern that separates information senders (editors) from information users (subscribers). Publishers publish various types of information in a shared space, which is increasingly becoming a public cloud these days. Subscribers present their interest in different types of information. When a publisher publishes an information item of a specific type, a broker finds subscribers who are interested in that type of information and passes on the information item to them. Increasingly, large IT providers such as IBM, Google and Microsoft are offering publish-subscribe functionality as a service. Specifically, the paper addresses the problem of terminal congestion by introducing a cooperative approach, based on the Publish-subscribe communications paradigm. As I will do in this work, they used discrete event simulation to simulate a typical intermodal terminal environment for road rail transport. But the code of simulations used in this case is the Ruby language and this is a dynamic, interpreted, reflective, object-oriented, general-purpose programming language.

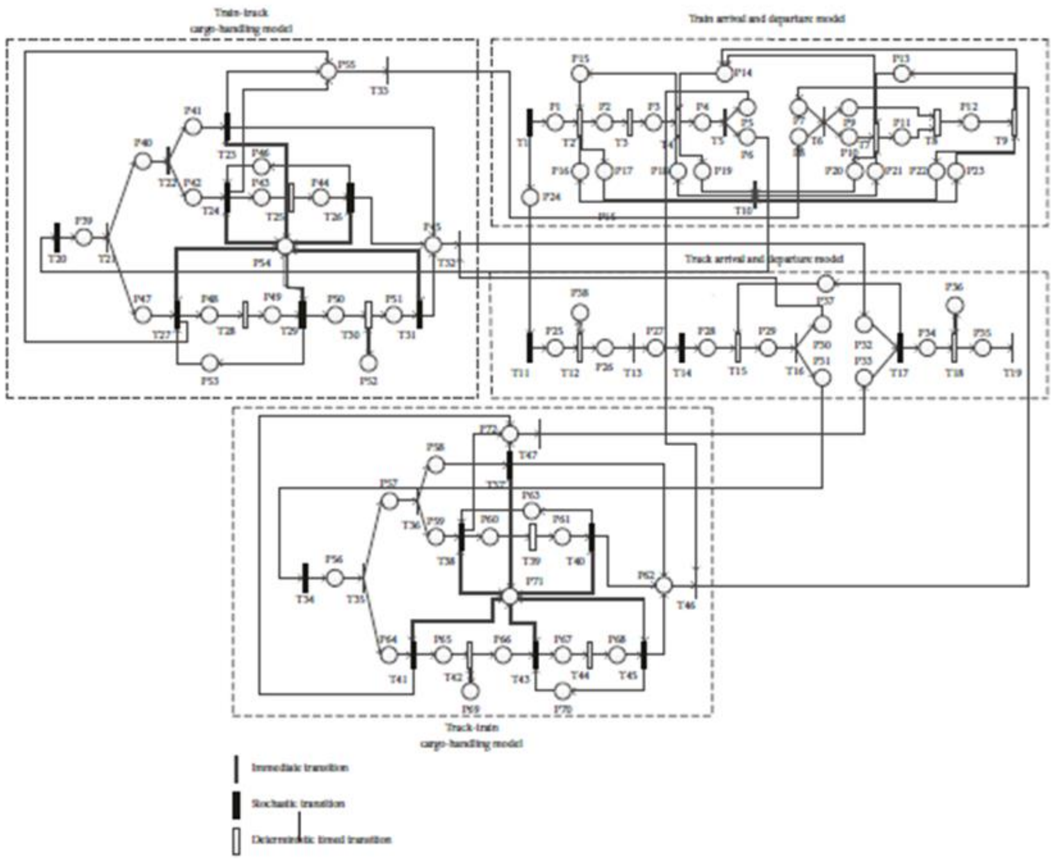
Another example of projects in transport engineering that include different software is the ViWaS (Viable Wagonload Production Schemes) project [13]. Rail transport of individual wagons or groups of wagons is an indispensable part of the transport chain. However, high production costs and low-quality levels have led to a continuous decrease in market shares in recent years. To counter this trend, ten European companies and research institutions, covering the fields of rail transport and logistics, have joined forces in the framework of the ViWaS research and development project. The objective: Innovative and practical solutions for the sustainable transport of freight wagons.

This project talks about Improved "last mile" operating concepts incorporating hybrid locomotives and bimodal shunting engines (by Bentheimer Eisenbahn, Fret SNCF and SBB Cargo with the support of HaCon and NEWOPERA): The new production method for last mile delivery is based on the idea of separating train movements and shunting processes from service tracks by using bimodal road and rail tractors. He also talks about Modular Wagon Technologies for a flexible and efficient use of resources (by Wascosa and SBB Cargo) and about Intelligent Wagon Telematics that allows better tracking at reduced costs (by Eureka). Finally, he talks about a new simulation tool for planning and optimising individual wagon networks (by ETH Zürich): WagonSIM. It is an agent-based system of rail freight network simulation tool to facilitate the optimization of SWL production schemes. It is based on the OpenSource software MatSIM. The tool models the layout of freight wagons according to the routes within the actual SWL network. Therefore, the modelling of two network levels, the production network and the physical infrastructure, is required. This project has the concept of innovation and the use of various software to address current problems in intermodal stations and intermodal transport in general, which I need as inspiration for the work to be done.

There are cases where the simulation of the intermodal terminal uses very different resources until now. Such is the case of the study of a multi-lane rail intermodal terminal (MYRIT) in China [14] where Petri Net is used, which is a widely used tool that uses graphic elements as a representation to describe the structure and dynamics of DESD, such as computer systems and manufacturing systems. The simulation model is based on the Timed Petri dish network (TPN). TPN is a bipartite diagram described by the five tuples as shown in the following formula:

TPN = $(P, T, \text{Pre}, \text{Post}, F)$

where P represents the set of places, T is the set of transitions, Pre is the pre-incidence matrix, Post is the post-incidence matrix and the function F specifies the time associated with each transition. There are three types of transitions used in this model: immediate transition, stochastic transition, and deterministic transition. Corresponding to MYRIT's operations, the TPN model is segmented into three sub-models: the train arrival and departure model, the truck arrival and departure model, and the load handling model. Of these three models, the provisions regarding the model of arrival and departure of trains are applicable to my thesis.



Train-truck cargo-handling model
Train arrival and departure model
Truck arrival and departure model
Truck-train cargo-handling model
Immediate transition
Stochastic transition
Deterministic timed transition

Figure 1 - TPN model of multiyards railway intermodal terminal example.

In addition to those mentioned above, there are other software, most of them subject to commercial license, for the simulation of intermodal terminals. We are faced with a range of possibilities, so what can new software bring to the topic in question, and what would be its advantages?

To begin with, I have not found any specific background on the use of SimEvents® in the simulation of intermodal terminals, but rather on the use of the software for more punctual issues such as:

- Analysis of traffic congestion queues generated in intermodal stations. [15]

The Consorzio Napoletano Terminal Containers (CO.NA.TE.CO.), located in the Port of Naples, is continuously exposed to a traffic congestion problem. A queuing model has been developed to analyse this congestion problem. The model includes part of the export process. The entities represented the vehicles accessing the terminal, and the servers were the gateway and the process of moving the cargo to the yard. Through this project it has been proven that all processes within the terminal were heavily dependent. The study also shows that the solution should consider the simultaneous reduction of service time at the access door and in the yard.

The project concludes that queuing model simulations are very useful for analysing a wide range of systems, such as the study. In addition, different scenarios have been modelled at a minimum cost to decide the best investment. This simulation can prevent unnecessary large investment companies that are usually carried out outside.

- Analysis of the times in the load flow of an intermodal network.[16]

In this case, we are talking about the Port of Seville. This allows you to switch from one mode of transport (rail) to another (sea) at the port's container terminal. Some constraints, such as the operational time slot restricting freight train access to the port within a certain time slot or the need to reverse the train before entering the port, cause significant delays in the intermodal chain. It is concluded that a temporal analysis of the process is necessary to check the critical points. The entire process is simulated from the moment the goods leave the station of origin by train until they leave the port of Seville by boat to their destination. To this end, a queuing network model was developed to simulate the travel time of the load. The database consists of daily freight train departures and daily ship departures (including berthing, berthing or loading/unloading times). The ultimate objective of this work is twofold: first, to provide a validated model of containerized cargo flow and, second, to demonstrate that such queue models can become a powerful tool to support future investment decisions.

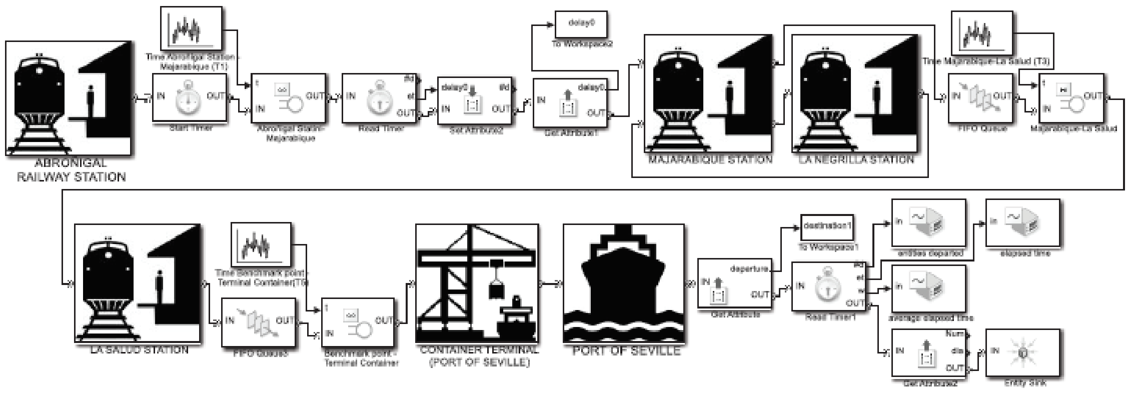


Figure 2 - Schematic itinerary of the model proposed here.

- Calculation of the capacity of a railway node, a passenger station. [17]

The study of the railway environment is dealt with here. Given its invariability from the conceptual point of view, it is considered very useful to be able to quickly verify its characteristics from the point of view of the analysis of a node, independently of the characteristics it will have with respect to the non-railway environment, for which the study is undoubtedly more varied and complex, and conditioned by the situation. Simulation makes it possible to considerably lighten the problem-solving phase of the node problem, which is precisely that of the plant capacity assessment; without the help and immediacy of simulation, the problem must in fact be tackled analytically, or through the empiricism of experience, while with the help of the simulation environment it is possible to immediately verify limits and possible variations of the implants with great flexibility and precision. This study focuses on the simulation of traveling nodes, but the method is applicable to any other type of node.

According to this work, the simulation of terminals in a computer environment, through the described method, is a viable alternative to the more known methods to solve the problem of the nodes in case a quick and explanatory preliminary analysis is required. He states that while other methodologies potentially allow for larger amounts of output data or greater accuracy, they require longer implant times and a deeper understanding of the operator. It also indicates that the method described above provides a quick and simple overview of the plant under examination, verifies the effects of changing its various parameters, both structurally and in terms of flow, or evaluates what may occur in the event of failures, delays or other inconveniences. It names the difficulties encountered and overcome during the study and states that these have produced a method that has its strong point in its elasticity and replicability, because, after the complex implementation process, a language is created that is universally applicable to each terminal, whether existing or under development or updating, and potentially replicable in any simulation environment matrix like that described in the project.

It can be concluded that the use of SimEvents® has been practically non-existent, this may be due to the relative complexity of having to program the entire system as opposed to the ease of use of specific intermodal terminal simulation software.

There are also jobs where another discrete event software has been used. In the case I will mention the software used is Witness, which has been used to simulate intermodal rail transport terminals with a classification zone.[18]

This project is part of a research line of the Area of Organization Engineering of the University Carlos III of Madrid. This line of research aims to study railway transport terminals.

Along these lines, the director of this project developed a simulation tool for her master's degree project that allowed her to study different terminals where containers were exchanged between trains and trucks using mobile cranes and gantry cranes. This tool consists of two elements: a simulation model programmed with Witness PwE 2.0 Manufacturing Performance Edition ("Witness"), which represents the generic design and operation of the terminals considered as object of study, and an MS Excel file, which allows the specific characteristics of the terminal to be simulated to be customized. One of the most interesting aspects of this tool is its flexibility, her than allows you to utilize on epitome of modelling without need of reschedule it, to scan on echelon of service, her output y on use of them facilities y resources of differing terminals.

We can name that Matlab® software is used in more papers on the subject, but as a simple calculation or design tool:

- For example, in the creation of algorithms to compare other existing ones. [19]

In this case, we talk about the design of a network with Internet Protocol (IP), which is a long-term project where many aspects must be taken into account. These include design costs, network sustainability and traffic engineering aspects. He clarifies that when designing an IP network for an industry such as seaports or container terminals, factors related to traffic engineering and scheduling need more attention. This thesis addresses design issues and suggests an IP network solution using the Open Shortest Path First (OSPF) routing algorithm concept.

Routing in OSPF telecommunication networks is determined by calculating the shortest paths with respect to the link weights established by the network operator. In this investigation, the shortest route has been calculated based on the current status, current location, distance and load limitation of the Automatically Guided Vehicles that use cassettes to transfer containers, which are called C-AGVs.

A MATLAB-based simulation tool is developed that is used to test and compare the OSPF algorithm with general programming processes in a container terminal. The results show that using the OSPF concept will result in higher productivity.

- And in container routing problems with stochastic time variables in an intermodal transport system [20].

In this project the authors have considered the problem of routing containers with stochastic time variables in a maritime-railway intermodal transport system. The problem is formulated as a randomly restricted integer binary programming model that includes stochastic travel times and stochastic transfer times, with the goal of minimizing the expected total cost. They have proposed two opportunity constraints to ensure that the container service meets the ship's obligations and the on-time delivery of cargo at pre-arranged probabilities. They have employed a hybrid heuristic algorithm to solve the randomized programming model of binary integer numbers. And they have carried out two case studies demonstrating the feasibility of the proposed model and analysing the impact of stochastic variables and the limitations of chance on the optimal solution and total cost.

Matlab's characteristic is its constant evolution, improving year after year its performance, it is enough to see two new versions published every year [21]. In this context, using this tool gives us the possibility to update our model constantly, with new programming options, improving its results, which is a great advantage over the use of specific software.

Another advantage is the compatibility of the simulation structure with others discrete event software. Given that there are many of these programs such as ProModel, which is a discrete-event simulation tool that also allows modelling of continuous processes; Enterprise Dynamics, a simulation software platform to model and analyse virtually any manufacturing, material handling and logistics challenge; AnyLogic, a general purpose multimethod modelling tool; Arena, a discrete event simulation program that also allows modelling of continuous processes; among others [22], it would allow us to use open source software for simulation, giving an open and global approach, where anyone interested in the subject, be it a student, a teacher, a professional or a curious person, can access and learn how it is, how it is designed, which are the variables, among other elements, of an intermodal terminal.

3. Hypotheses and design options.

In the following chapter I will begin by describing the general operation of an intermodal terminal, the parameters to be considered, and the different subsystems in which it can be differentiated, and the interrelationships that exist between them. I will now discuss about the various conditions that I have taken into account for the design of the simulation of each of these subsystems, with their corresponding hypotheses and conditioning factors. To conclude, I will make some clarifications about the chosen model and its possibility of expansion.

3.1. Essential elements of an Intermodal Terminal.

We can list five essential group of elements of an Intermodal terminal:

- The first are the Tracks, where we have three different kind of tracks: Operating tracks, accessible to the cranes, Stopping and manoeuvring tracks, where loaded trains are left waiting for departure or empty before arrival at the terminal, and pick-up and drop-off tracks, are intended to enable technical stops to be made for changing locomotives, checking loading and checking documents.
- In the second group are the Areas for the movement of road vehicles and lifting equipment, and again, we have three different kind of areas: Loading/unloading lane, where ITU exchanges between road and rail vehicles take place, Overpass lane, are designed to allow a road vehicle to overtake those awaiting loading/unloading and has the same direction of movement, and Manoeuvring area of road vehicles, it can be a round pitch for reversing the direction of travel.
- The next group is only for the Storage area, is the place where the ITUs are deposited, preferably depending on the type and destination, pending their departure.
- The fourth group is for Technical warehouses for terminal functions (intervention vehicles, spare parts depots, etc.), workshops, washing areas and repair depots for the exclusive use of the terminal operating vehicles and not for the ITUs.
- And the last group is for the area for gates and related operations, offices, sanitary facilities. At the gate of a terminal are carried out certain activities that can affect overall productivity as they are: check-in, i.e. delivery of documents and exchange of information on operations to be carried out, and control of the vehicle and ITUs.

From a functional point of view, the terminal consists of three sub-systems:

- The railway subsystem,
- The truck subsystem,
- The intermodal subsystem,

which operate and integrate in the terminal area.

The following figure shows a functional scheme of the sub-systems constituting an intermodal terminal.

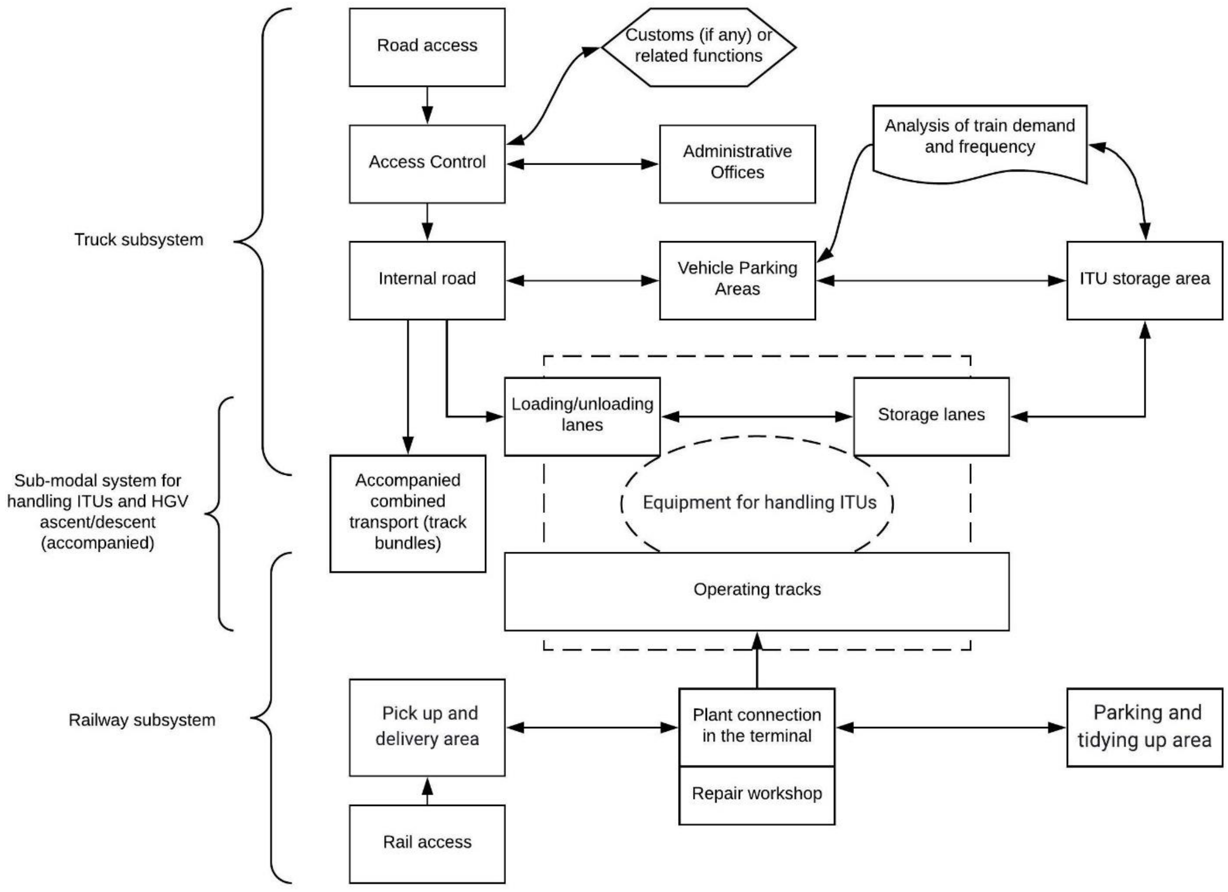

Truck subsystem
Sub-modal system for handling ITUs and HGV ascent/descent (accompanied)
Railway subsystem
Road access
Customs (if any) or related functions
Access Control
Administrative Offices
Analysis of train demand and frequency
Internal road
Vehicle Parking Areas
ITU storage area
Loading/unloading lanes
Storage lanes
Accompanied combined transport (track bundles)
Equipment for handling ITUs
Operating tracks
Pick up and delivery area
Plant connection in the terminal
Repair workshop
Parking and tidying up area
Rail access

Figure 3 - Functional scheme of the terminal sub-systems

Rail and road flows are variable and different from each other: a regular schedule of arrival and departure times of trains is contrasted by a wide variability in arrivals and departures of road vehicles. Generally, in fact, many road vehicles wait for the arrival of the train to be able to receive the ITUs with direct transhipment, while others receive the load from the parking and storage areas. If the train can remain on the operating tracks for many hours, the possibility of direct loading and unloading is much higher but at the same time the terminal capacity is reduced for track occupation.

If the train arrives in the early morning hours and stays on the operating track for the whole day until its departure in the evening, the incoming road vehicles determine the sequence of direct loading and unloading operations: it can be said that in this case the road vehicle is preferred because it has a very fast cycle. This priority for the lorry is characteristic of combined road-rail transport.

3.2. Design options.

The railway subsystem shall be designed with the implication that there shall be no delays in the truck subsystem or in the storage area that would impair the operability of the trains.

To work on my design, I will assume a specific layout of the infrastructure of an intermodal station (Figure 4), which will consist of:

- Three railways track of pick-up and delivery: A, B and C.
- Three operational railways track: 1, 2 and 3.
- A by-pass for shunting locomotives and crossings for the storage of broken ITUs: Z.
- Gantry cranes will be used.

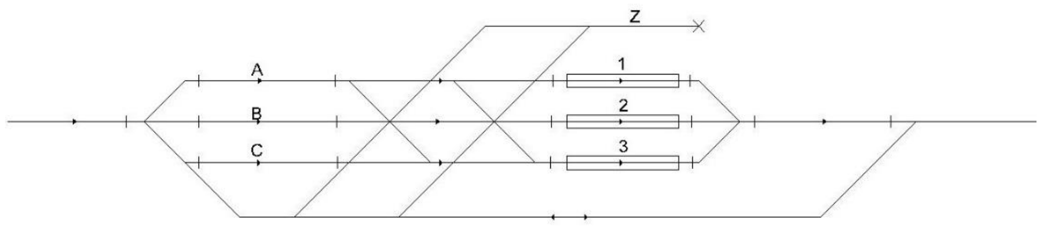


Figure 4 - Schematic view of the station

Figure 5 shows the profile to be used in the simulation, which consists of three main tracks, a truck loading/unloading lane, a truck overpass lane and three storage areas:

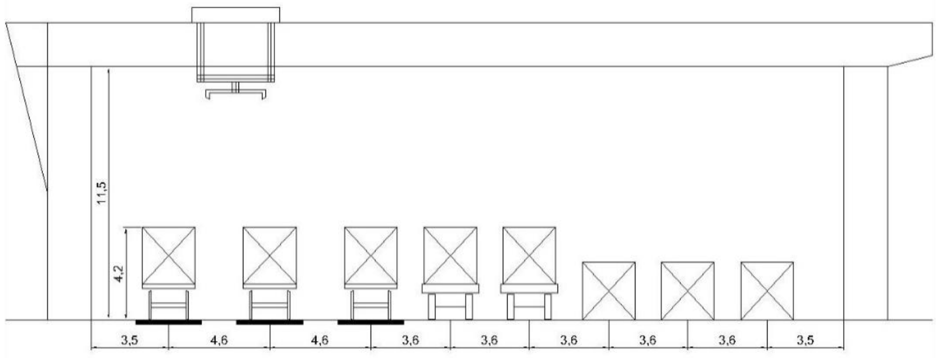


Figure 5 - Station profile diagram

I should mention that this profile, with its layout and distances, mainly affects the crane's service times. All this will be explained in detail in the section "4.4. Model calibration parameters."

Work will be carried out on five trains, each with approximately one hundred and ten (110) ITUs, according to the infrastructure hypothesis used for the daily work. Considering an average value of 250,000 t/year per freight train, the total for the 5 trains is 1,250,000 t/year, less than 1.5 Mt/year, so the profile chosen for the simulation, of three operational tracks and two portal cranes, is consistent with the recommended profile.[23]

3.3. About the model.

It is obvious that the model chosen is a very small and simple terminal, which does not mean that it cannot be extended in most aspects:

- Can be add more tracks of different types,
- The profile can be changed by adding lanes for different uses of trucks and more storage areas.
- Even internal train paths can be added and changed.
- Can be also add and connect the truck sub-system

All this serves to meet the real requirements of a true intermodal station.

4. Modelling.

In this section I will start with a brief description of SimEvents, which is the software package inside MatLab, and next to Simulink, which I will use. To continue with detailed descriptions of the different phenomena involved within the model. I will immediately continue with the detailed description of the whole model, work that I will divide in parts for a more pleasant reading improving its comprehension. The different parameters considered for model calibration will then be explained. Finishing with the sample of the output parameters that can be obtained with the modelling. I have tried to be as accurate as possible to facilitate reproduction.

4.1. SimEvents®.

As I have anticipated, the simulation is performed in the Matlab® software, more precisely in the Simulink® SimEvents® package [24]. I must clarify that the version I have used is the R2017b, since there are very deep changes with respect to previous versions.

SimEvents provides a discrete-event simulation engine and component library for analysing event-driven system models and optimizing performance characteristics such as latency, throughput, and packet loss. Queues, servers, switches, and other predefined blocks enable you to model routing, processing delays, and prioritization for scheduling and communication.

With SimEvents, can be study the effects of task timing and resource usage on the performance of distributed control systems, software and hardware architectures, and communication networks. You can also conduct operational research for decisions related to forecasting, capacity planning, and supply-chain management.

Among its Key Features are:

- Discrete-event simulation engine for multidomain system models.
- Entities with custom data attributes representing tasks, packets, and items.
- Blocks for queuing, service, routing, resource management, multicasting, replication, and batching.
- Statistics generation for delay, throughput, average queue length, and other metrics.

SimEvents software incorporates discrete-event system modelling into the Simulink time-based framework, which is suited for modelling continuous-time and periodic discrete time systems. In time-based systems, state updates occur synchronously with time. By contrast, in discrete-event systems, state transitions depend on asynchronous discrete incidents called events.

In a Simulink model, you typically construct a discrete-event system by adding various blocks, such as generators, queues, and servers, from the SimEvents block library. These blocks are suitable for producing and processing entities, which are abstractions of discrete items of interest.

One or more discrete-event systems can coexist with time-based systems in a Simulink model. This coexistence facilitates the simulation of sophisticated hybrid systems. You can pass signals from time-based components/systems to and from discrete-event components/systems modelled with SimEvents blocks. The combination of time and event-based modelling facilitates the simulation of large-scale systems that incorporate Discrete-Event Simulation in Simulink Models smaller subsystems from multiple environments.

Discrete-event simulations typically involve discrete items of interest. By definition, these items are called entities in SimEvents software. Entities can pass through a network of queues, servers, gates, and switches during a simulation. Entities can carry data, known in SimEvents software as attributes. In a discrete-event simulation, an event is an observation of an instantaneous incident that may change a state variable, an output, and/or the occurrence of other events. Events can correspond to changes in the state of an entity.

Resources are commodities shared by entities in your model. They are independent of entities and attributes and can exist in the model even if no entity exists or uses them. Resources are different from attributes, which are associated with entities and exist or disappear with their entity. [25]

4.2. Subsystems present in the model.

The general model I have assembled for later simulation can be divided into the following subsystems:

1. Arrival of the trains at the station at a set time.
2. After the trains arrive at the pick-up and drop-off tracks where they will remain until the actions of changing locomotives, documentation control, control of the general state of the wagons and until they can continue to their corresponding Operational tracks. In the event of the presence of wagons in poor condition, they shall be separated from the convoy.
3. Passage of trains from the pick-up and drop-off tracks of origin to the Operational tracks of destination, where, depending on the type of infrastructure, the origin and destination, a number n of trains may pass simultaneously (or not).
4. The trains arrive at the Operational tracks where, depending on the individual destination of each ITU present in each train, they will be unloaded, the ITUs will be exchanged between the trains present in the different Operational tracks and the corresponding ITUs will be loaded from the trucks and from the depot, all this using the service cranes.
5. Once this process has been completed, each train will have two paths to follow:

a. The first, when it is fully served, is to continue to the departure area to leave the station where, in addition, a final check of papers and loaded ITUs will be carried out;
b. The second, when it still needs to make exchanges of ITUs with other trains that were not present in the Operational tracks at the time, is to return to the pick-up and drop-off tracks to give way to these trains, but with the condition of not passing again through the controls of documentation or broken wagons, restarting the process.

4.3. Model simulation.

I will divide the simulation into seven parts for a more detailed, but less cumbersome explanation:

1. Generation.
2. Resources.
3. Pick-up and delivery rails.
4. Train crossing.
5. Operational tracks.
6. Returns and Exits.
7. Global explanation of the simulation.

4.3.1. Generation.

After analysing the possibility of generating each ITU of each train as an individual entity and seeing that this was not compatible with the system I had proposed and placing three generating blocks (one for each track), I decided to generate the Train entities through a single block called “Entity Generator” and represent the number of ITUs as attributes of these entities. As a single block I have more control over the arrival times and I create a better approximation of reality by using an Output Switch with the "First port that is not blocked" option to distribute the trains on the different pick-up and drop-off tracks.

To be able to assign an entry time for each train, the generation is carried out through a pattern, an option that is incorporated in the same block "Insert Pattern ...", in the "Entity generation" tab.

To facilitate tracking and to be able to apply the conditions specific to each train, I listed them by adding the ID attribute through the following command lines in the "Event actions" tab:

```
persistent next_id
if isempty(next_id)
    next_id = 1;
    % Set seed for random number generator
    rng(12345);
end
```

```
% Assign ID attribute
entity.ID = next_id;
next_id = next_id + 1;
```

I have also added more attributes: "Origin", which indicates the Pick-up and drop-off track of the train; "Destination" indicates the Operational track corresponding to the train; "BrokenWagons", which indicates the presence or not of bad wagons; "ServiceTimeBrokenWagons", a time parameter necessary for the service in the event that there are broken wagons and "Cycle" which counts the number of times the train has passed through the operational tracks.

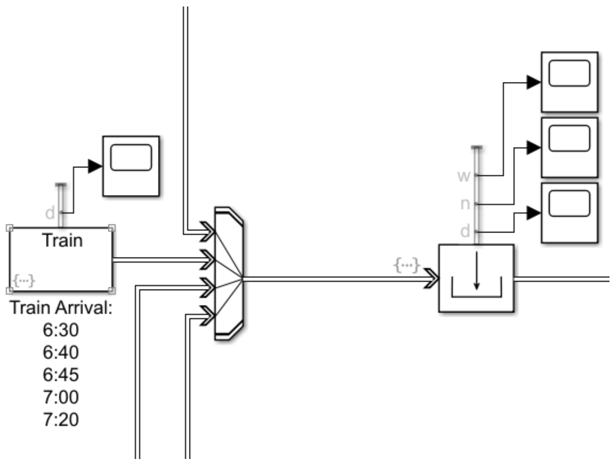


Figure 6 – Generation

4.3.2. Resources.

In this version of the software we have a very useful block for my purposes: the "Resource Pool" block, which allows me to easily control the number of cranes and locomotives available in my system, and through the "Resource Acquire" and "Resource Release" blocks to decide when to use and when to stop using each resource.

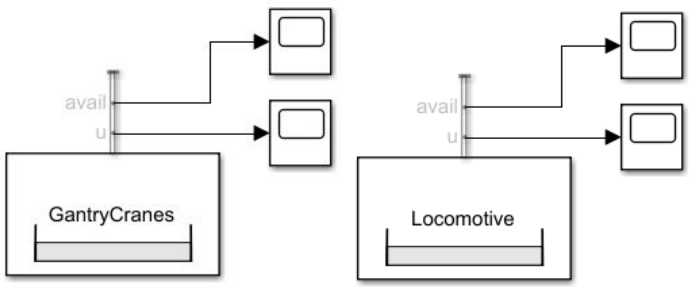


Figure 7 – Resources

4.3.3. Pick-up and drop-off tracks.

Starting from a basic configuration of the Gate-Server-Function type where Gate and Function allow me to determine under what conditions a train enters this track and Server the time it will remain on it, I can simulate the entry and the time taken to control the train.

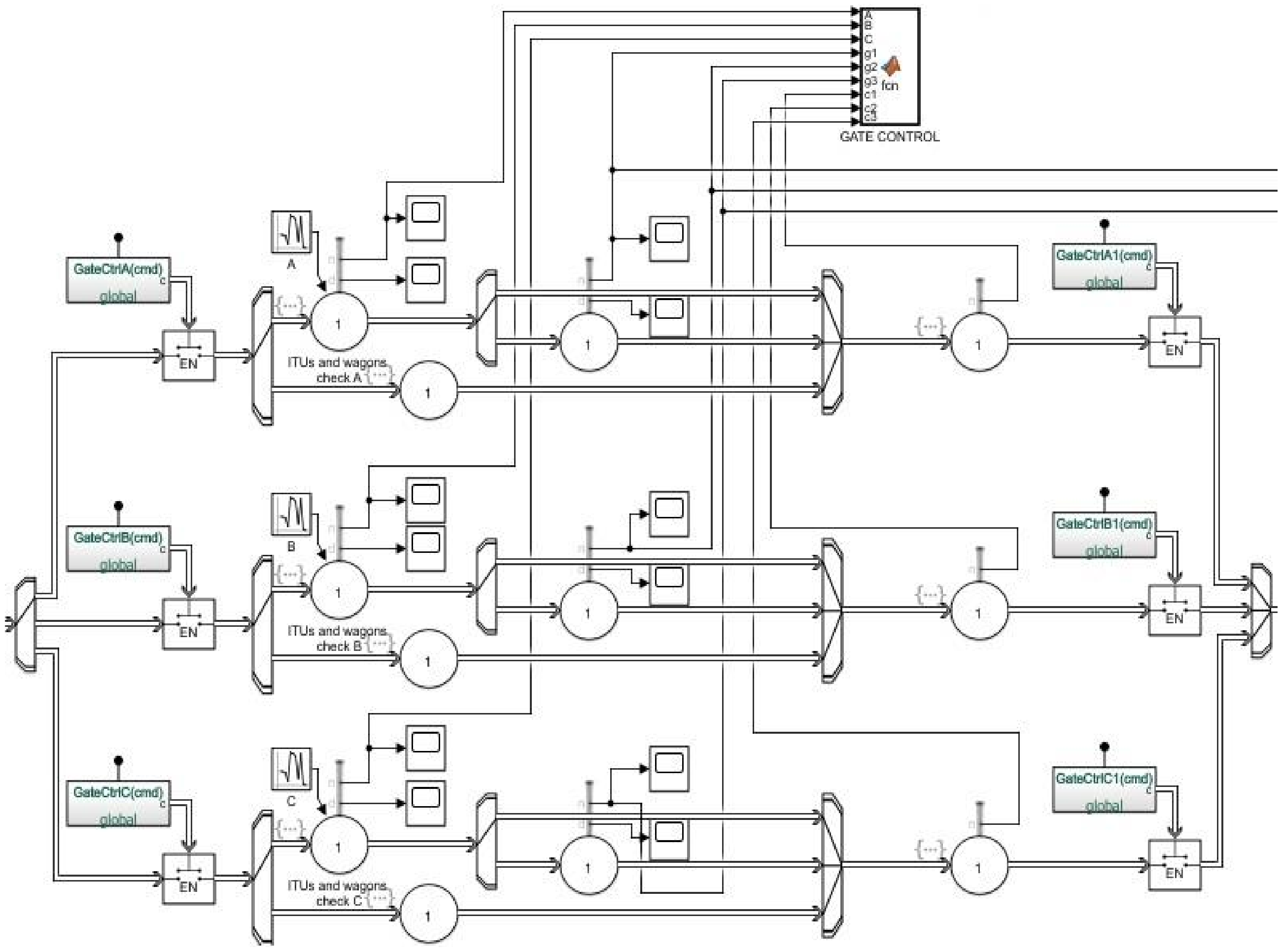
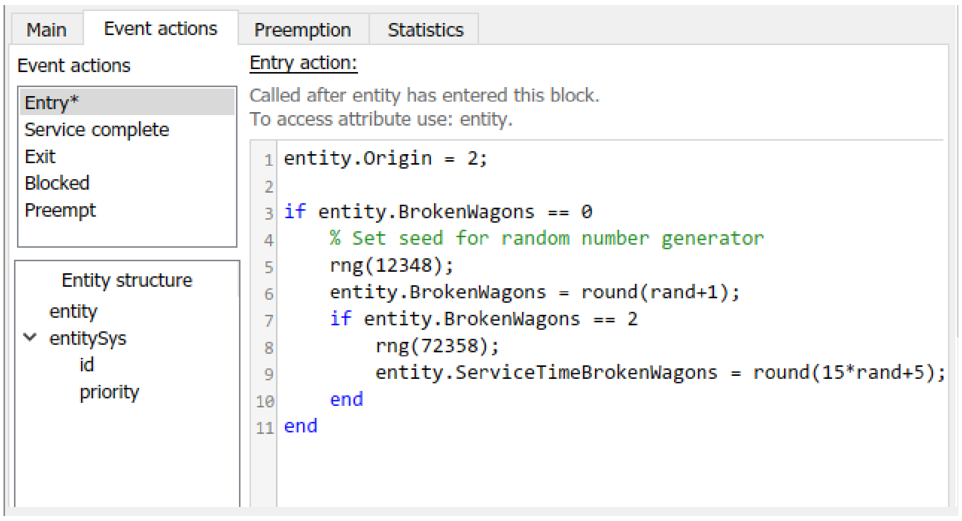


Figure 8 - Pick-up and delivery rails

As shown in the figure, I have used blocks of the type "Output Switch" with the "From attribute" option choosing "Cycle" as the attribute which defines whether the entity has already travelled this way or not, in case the entity advances to a server used only to modify the "Origin" variable. If not, however, it moves on to the "Entity server" block where the entity is "served" for a certain period by a random number with a Gaussian distribution. In addition, the presence of broken wagons is randomly generated and the time it would take to separate these wagons and take them to storage. For the latter, the command lines shown in figure 9 have been used.



Entry action:

Called after entity has entered this block.
To access attribute use: entity.

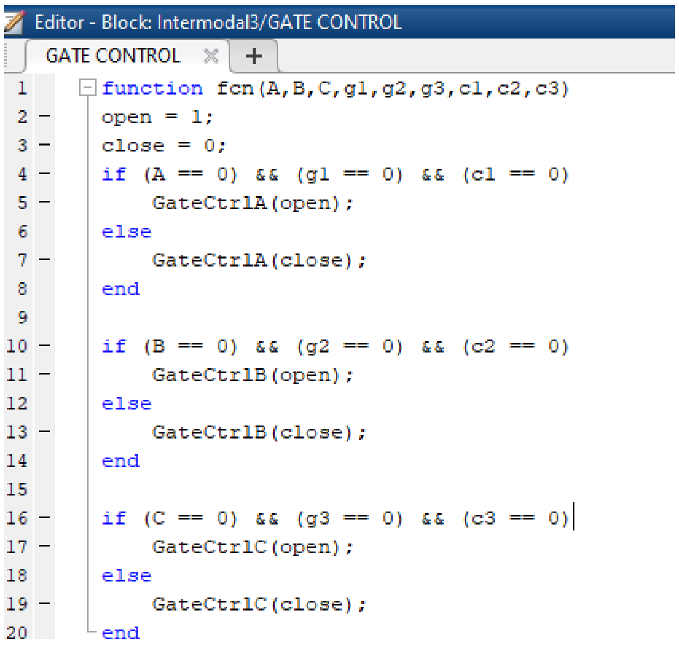
```
entity.Origin = 2;

if entity.BrokenWagons == 0
    % Set seed for random number generator
    rng(12348);
    entity.BrokenWagons = round(rand+1);
    if entity.BrokenWagons == 2
        rng(72358);
        entity.ServiceTimeBrokenWagons = round(15*rand+5);
    end
end
```

Figure 9 - Service Time Broken Wagons

The first line changes the value of the Origin attribute according to the corresponding path. Analysing them in detail the others we see that they assign a value 1 or 2 to the BrokenWagons attribute: 1 means the absence of broken cars and 2 the presence of them, modifying in this case also the ServiceTimeBrokenWagons attribute between the values 5 and 20 minutes. These attributes are then used in the following output switch and entity server.

This string ends with a last server block used to transmit the value of the "Destination" attribute that we will use later, and a last Gate type block.

The Matlab Function block uses the occupancy values of the various servers plus conveniently imposed conditions to control the opening of the Gates, as shown below in figure 10.



```
function fcn(A,B,C,g1,g2,g3,c1,c2,c3)
open = 1;
close = 0;
if (A == 0) && (g1 == 0) && (c1 == 0)
    GateCtrlA(open);
else
    GateCtrlA(close);
end

if (B == 0) && (g2 == 0) && (c2 == 0)
    GateCtrlB(open);
else
    GateCtrlB(close);
end

if (C == 0) && (g3 == 0) && (c3 == 0)
    GateCtrlC(open);
else
    GateCtrlC(close);
end
```

Figure 10 - Gates condition

4.3.4. Train crossing.

This section is also governed by the combination Gate-Server-Function.

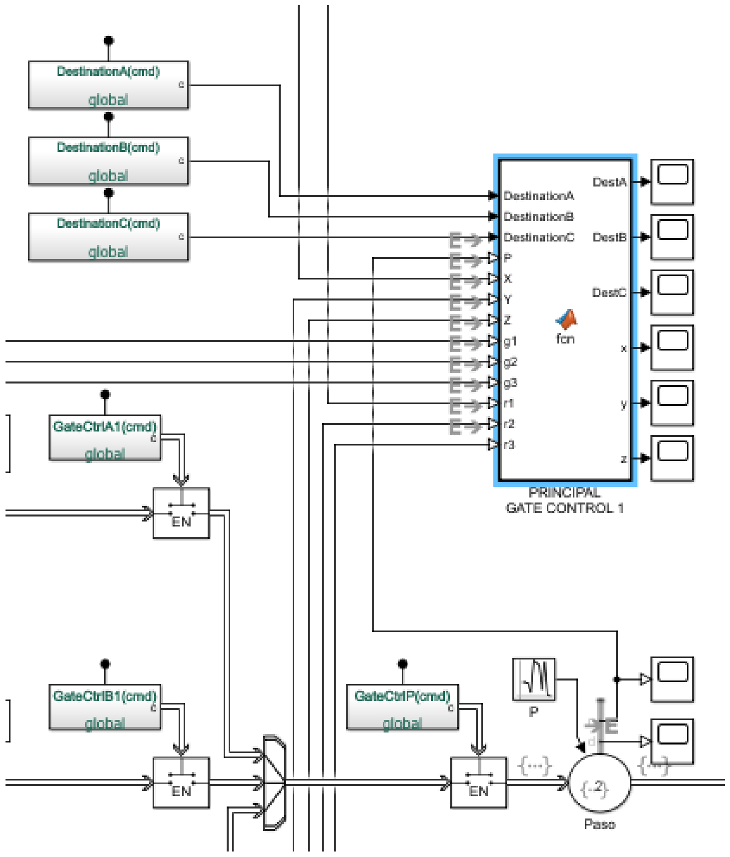


Figure 11 - Train crossing

In this segment I want to represent the existing infrastructure. There may be several cases, for example the following:

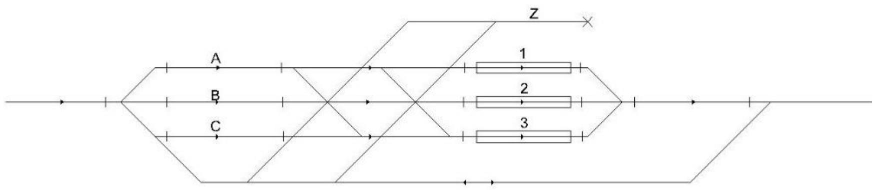


Figure 12 - Schematic view of the station

From this I have created a matrix of values 0 and 1, where the possibility of making different routes at the same time is compared. 0 indicates the impossibility of making them. Instead 1 tells us that the two routes are compatible simultaneously. For example, the following matrix represents the above-mentioned infrastructure.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Matrix 1

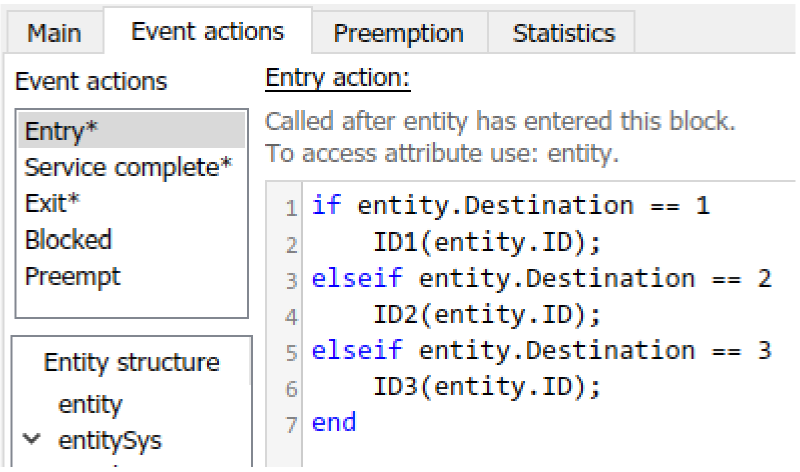
As an example, we analyse the element in row 1, column 5, is a 1, this tells us that the trip A1 (from A to 1) is compatible with the B2 trip. Another example, from row 6, column 7, is a 0, this tells us that trip B3 is not compatible with trip C1. This matrix actually represents the routes compatibility matrix of the interlocking systems which manages the shunting yard.

Using the values of the Destination attribute and knowing the origin track of each train as it is served, I can determine whether the routes are shared.

As it is a very extensive code, the writing inside the “Matlab function” block I add it in the appendix. It is enough to clarify here that the signals of the broken car servers are used, due to the movement that the locomotives, the operating routes and the returns must carry out.

Inside the "Entity Server" block I had to add several commands for the correct functioning of the program:

- The first is used to transmit the ID of the train being served according to its destination (Figure 13).



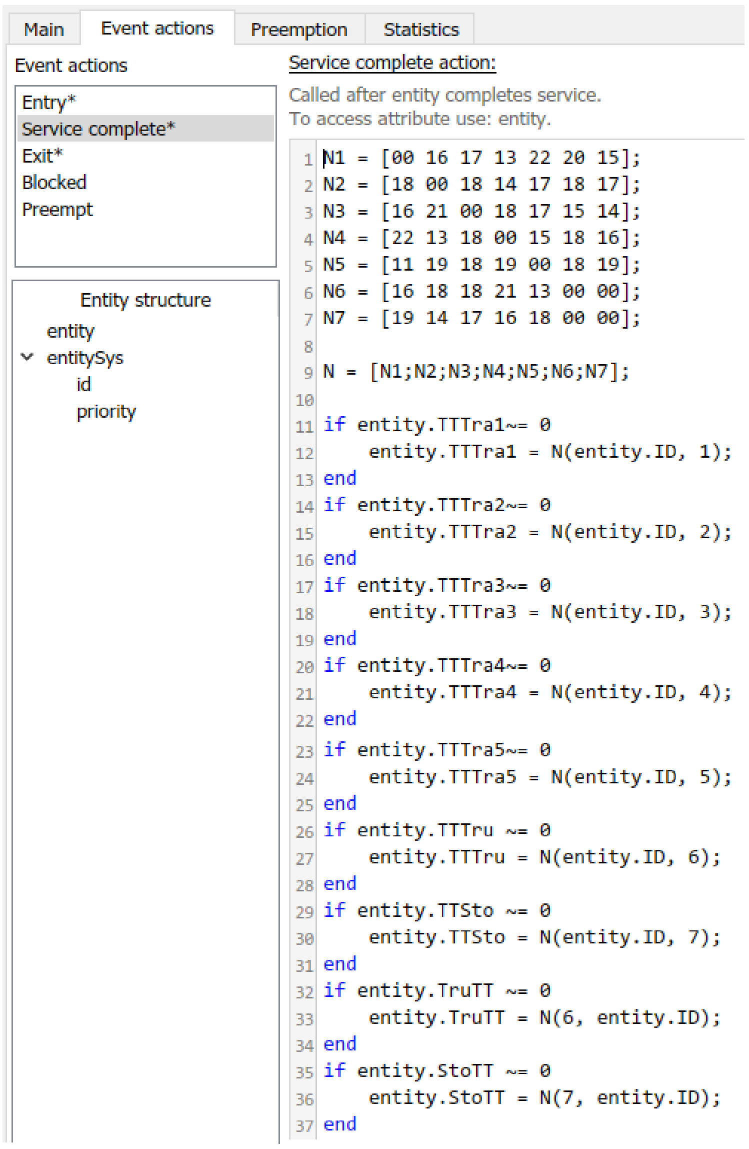
Entry action:

Called after entity has entered this block.
To access attribute use: entity.

```
if entity.Destination == 1
    ID1(entity.ID);
elseif entity.Destination == 2
    ID2(entity.ID);
elseif entity.Destination == 3
    ID3(entity.ID);
end
```

Figure 13 - How transmit the ID

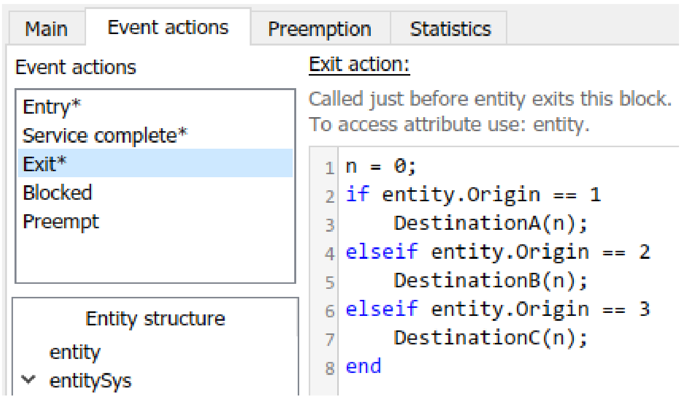
- The second is to assign the corresponding Origin-Destination values (Figure 14).



```
N1 = [00 16 17 13 22 20 15];
N2 = [18 00 18 14 17 18 17];
N3 = [16 21 00 18 17 15 14];
N4 = [22 13 18 00 15 18 16];
N5 = [11 19 18 19 00 18 19];
N6 = [16 18 18 21 13 00 00];
N7 = [19 14 17 16 18 00 00];

N = [N1;N2;N3;N4;N5;N6;N7];

if entity.TTTra1~= 0
    entity.TTTra1 = N(entity.ID, 1);
end
if entity.TTTra2~= 0
    entity.TTTra2 = N(entity.ID, 2);
end
if entity.TTTra3~= 0
    entity.TTTra3 = N(entity.ID, 3);
end
if entity.TTTra4~= 0
    entity.TTTra4 = N(entity.ID, 4);
end
if entity.TTTra5~= 0
    entity.TTTra5 = N(entity.ID, 5);
end
if entity.TTTru ~= 0
    entity.TTTru = N(entity.ID, 6);
end
if entity.TTSto ~= 0
    entity.TTSto = N(entity.ID, 7);
end
if entity.TruTT ~= 0
    entity.TruTT = N(6, entity.ID);
end
if entity.StoTT ~= 0
    entity.StoTT = N(7, entity.ID);
end
```

Figure 14 -How assign the corresponding Origin-Destination values

- Finally, the third is used to reset the value of the "Origin" signal as appropriate (Figure 15).



```
n = 0;
if entity.Origin == 1
    DestinationA(n);
elseif entity.Origin == 2
    DestinationB(n);
elseif entity.Origin == 3
    DestinationC(n);
end
```

Figure 15 - How reset the value of the "Origin" signal

4.3.5. Operational tracks.

For this section, a basic Gate-server-function system was started for each lane. This system turned out too simple and as the simulation progressed, the proposed system could not define vital things such as different service times depending on whether the ITU was going to a truck, a depot or another train. An "Entity Replicator" block was added to work with each type of movement independently. For this example, ten exits are counted:

- **5** for trains: exchanges between current train and train 1, current train and train 2, current train and train 3, current train and train 4 and current train and train 5. There will be no problem if the current train coincides with train n because the trip matrix attributes a zero value to this case.
- **2** for train-truck and truck-train exchanges.
- **2** for train-tank and train-tank exchanges.
- **1** to continue with the simulation.

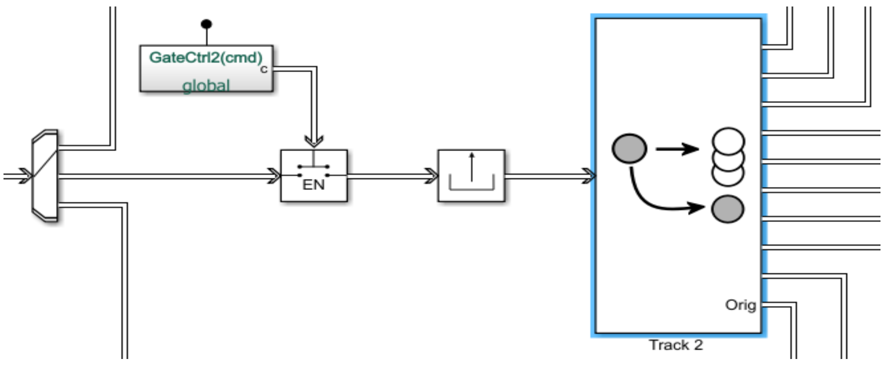


Figure 16 - Entity Replicator

Each Entity Replicator output is followed by a series of blocks, which will be different in the case of exchanges between trains compared to exchanges between different modes.

For the first five departures, the exchange of ITUs **between trains** is done as shown in Figure 17.

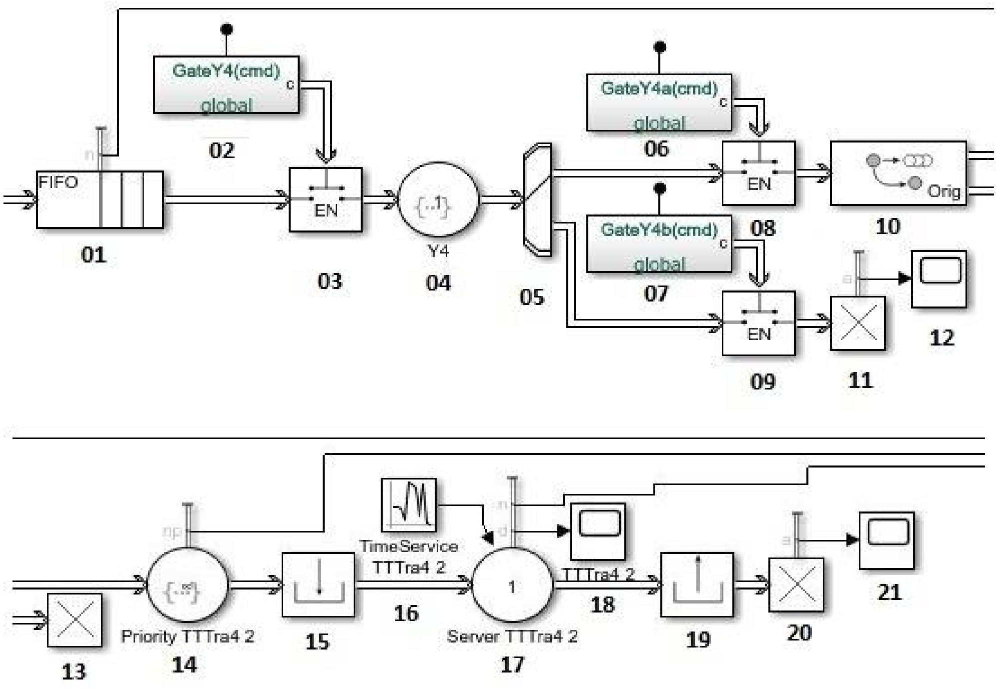
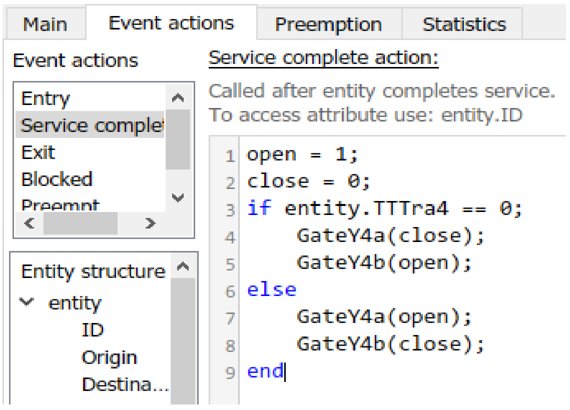


Figure 17 - Operational rails between trains

Where the blocks listed are as follows:

- **01 - "Entity Queue":** its use is only in cases where the corresponding train with this subsystem, in this example the exchange with the ID 4 train, is not physically available for the exchange. It's a question of design.
- **02, 06, 07 - "Simulink Function":** element used to control the opening of Gate blocks.
- **03, 08, 09 - "Entity Gate":** for each specific case, the corresponding entity may or may not pass through.
- **04 - "Entity Server":** Used to handle Gates 08 and 09 according to the corresponding attribute. The command lines shown in figure 18 have been used.



Service complete action:

Called after entity completes service.
To access attribute use: entity.ID

```
open = 1;
close = 0;
if entity.TTTra4 == 0;
    GateY4a(close);
    GateY4b(open);
else
    GateY4a(open);
    GateY4b(close);
end
```

Figure 18 - Gates control

- **05 - "Output Switch":** in conjunction with entity server 07, they avoid the zero-attribute error for the entity replicator block below.

- **10 - "Entity Replicator":** a key element. It generates as many identities as ITUs I must serve. This is done through the corresponding attribute, in this case "TTTra4" (Train to Train 4).
- **11, 13, 20 - "Entity Terminator".**
- **12, 18, 21 - "Scope":** used to display different statistics.
- **14 - "Entity Server":** used to give different priorities when serving trains. In this case the priority is high and is assigned a value of 100.

Kind of interchange	Priority	Priority Value
Train to Train	High	100
Train to Track	Medium	200
Train to Storage	Medium	300
Track to Train	Low	400
Storage to Train	Low	500

Table 1 - Priority values

- **15 - "Resource Acquire":** Here you simulate the occupation of one of the cranes to start the loading/unloading process.
- **16 - "Random Number":** Generates a random number within a Gaussian distribution with average and variance determined for each case. It was also decided to change the Seed, to have different values between the cases that coincide in mean and variance.
- **17 - "Entity Server":** Simulates the operating time for each crane shot.
- **19 - "Resource Release":** simulates the inoccupancy of the corresponding crane, once the work is finished.

In the next four outputs, the exchange of ITUs **<u>between different modalities</u>** is done as shown in Figure 19.

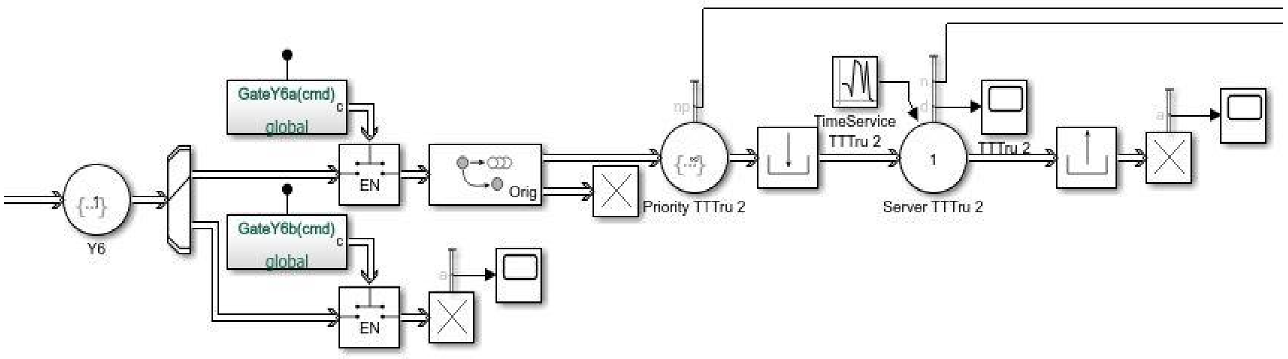


Figure 19 - Operational rails between different modalities

We can see that they are the same blocks, but the numbers 01, 02 and 03 are missing, because they are used when the exchange between trains is not possible because a train is not physically located.

The last output is done as shown in Figure 20.

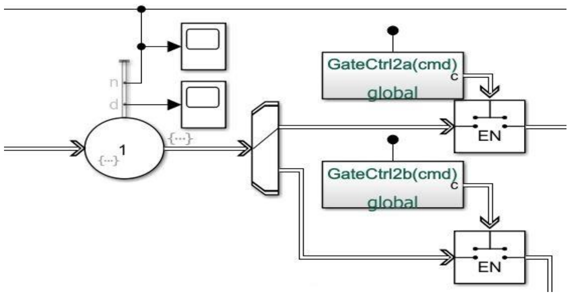
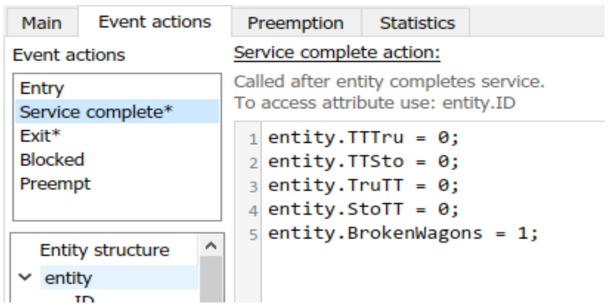


Figure 20 - The last output

Where we see an "Entity server" block used to avoid duplication of work if the served train must return to continue its exchanges. This is done through the command lines shown in Figure 21.



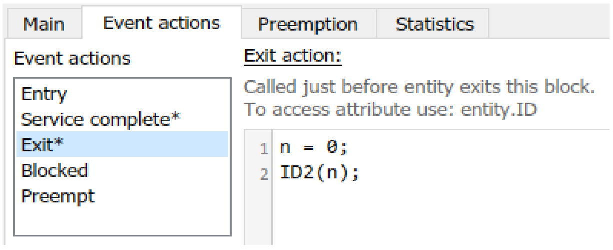
Service complete action:

Called after entity completes service.
To access attribute use: entity.ID

```
entity.TTTru = 0;
entity.TTSto = 0;
entity.TruTT = 0;
entity.StoTT = 0;
entity.BrokenWagons = 1;
```

Figure 21 - Change values

It is also used to reset the value of the ID1, ID2 or ID3 signals as appropriate.



Exit action:

Called just before entity exits this block.
To access attribute use: entity.ID

```
n = 0;
ID2(n);
```

Figure 22 - Reset value ID

Then we see a set of one “Output Switch”, two “Gates” and two “Matlab Functions”. This assembly is responsible for sending the train to or from the train via the 'cycle' attribute.

4.3.6. Returns and Exits.

This section complements the last part of the previous section. If the train current still contains UTIs and must leave the place for a new train, the **return phase** will proceed. This phase is represented by a “Gate” block, a “Resource Acquire” (locomotive) block, an “Entity Server” block with its corresponding “Random Number”, a “Resource Release” and finally an

“Input Switch”, the same one shown in the generation section, to channel the trains to the beginning.

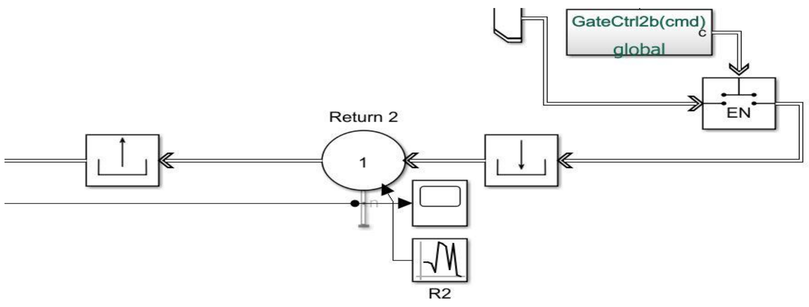


Figure 23 - Return phase

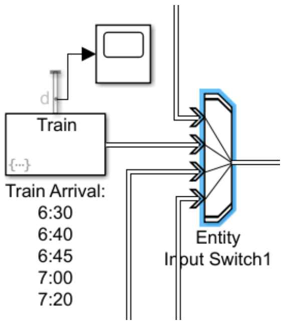


Figure 24 - Input Switch of Return phase

In the event that the train has been fully serviced, the **departure phase** will begin:

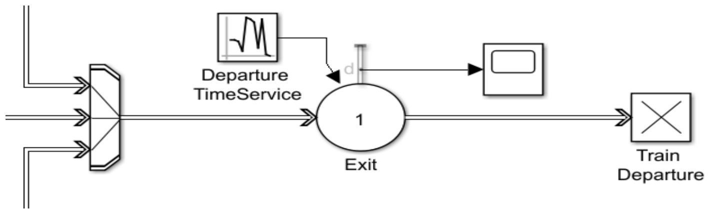


Figure 25 - Departure phase

Leaving the corresponding "Gate" named in the previous section, a new "Input Switch" is added, consisting of three inputs, one for the track, then the entities continue to an "Entity Server", with its corresponding "Random Number", finally arriving at an "Entity Terminator".

This entire system is controlled by a "Matlab Function" block called " Principal Gate Control 2 ". It receives multiple signals from most of the "Entity Server" blocks to control the entry and exit of trains accordingly. Despite receiving a very large number of signals, the logic used is very simple: When all the corresponding servers are empty, the corresponding action will be allowed. To see in detail this, please go to the second appendix where the corresponding command lines are.

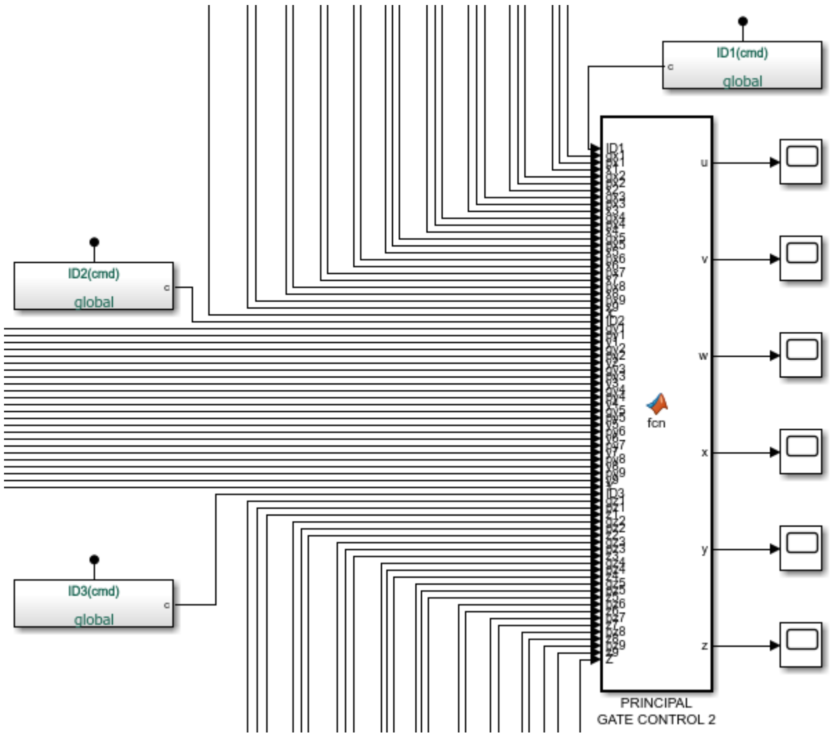


Figure 26 - Principal Gate Control 2

4.3.7. Global explanation of the simulation.

The simulation begins in the generation of the entities. Here, trains are created according to an arrival schedule that can be varied to study delays. It is immediately connected to the "Input Switch", which also receives the trains in the return phase.

The next step is the acquisition of a resource (locomotive) by entity, which represents the exchange of electric locomotive to diesel. The train entity shall continue to the first unoccupied track, where entry control shall be carried out. Here, things can happen, such as whether the train has wagons broken and it is the first or second time it has been around this place. According to each value is the path it will travel.

The train will remain in this section until the corresponding destination is free, and there are no other trains running, unless the two journeys are compatible.

Once the corresponding destination is free, the train will move towards it. Upon arrival, it will experience all the above tell in section five. Its will replicate the entity as many times as necessary.

It will be possible to start the exchange of loads between the train and trucks, and the train and storage, as we have hypothesized that there will be no delays in these two sectors. What will not be possible is the exchange between trains that are not available, as it would be in reality.

Once the corresponding exchanges have been completed, the train will proceed to return to the stopping and manoeuvring tracks, or to withdraw definitively if it has already been fully served, leaving room for a new train, which will not enter until the first train has left completely.

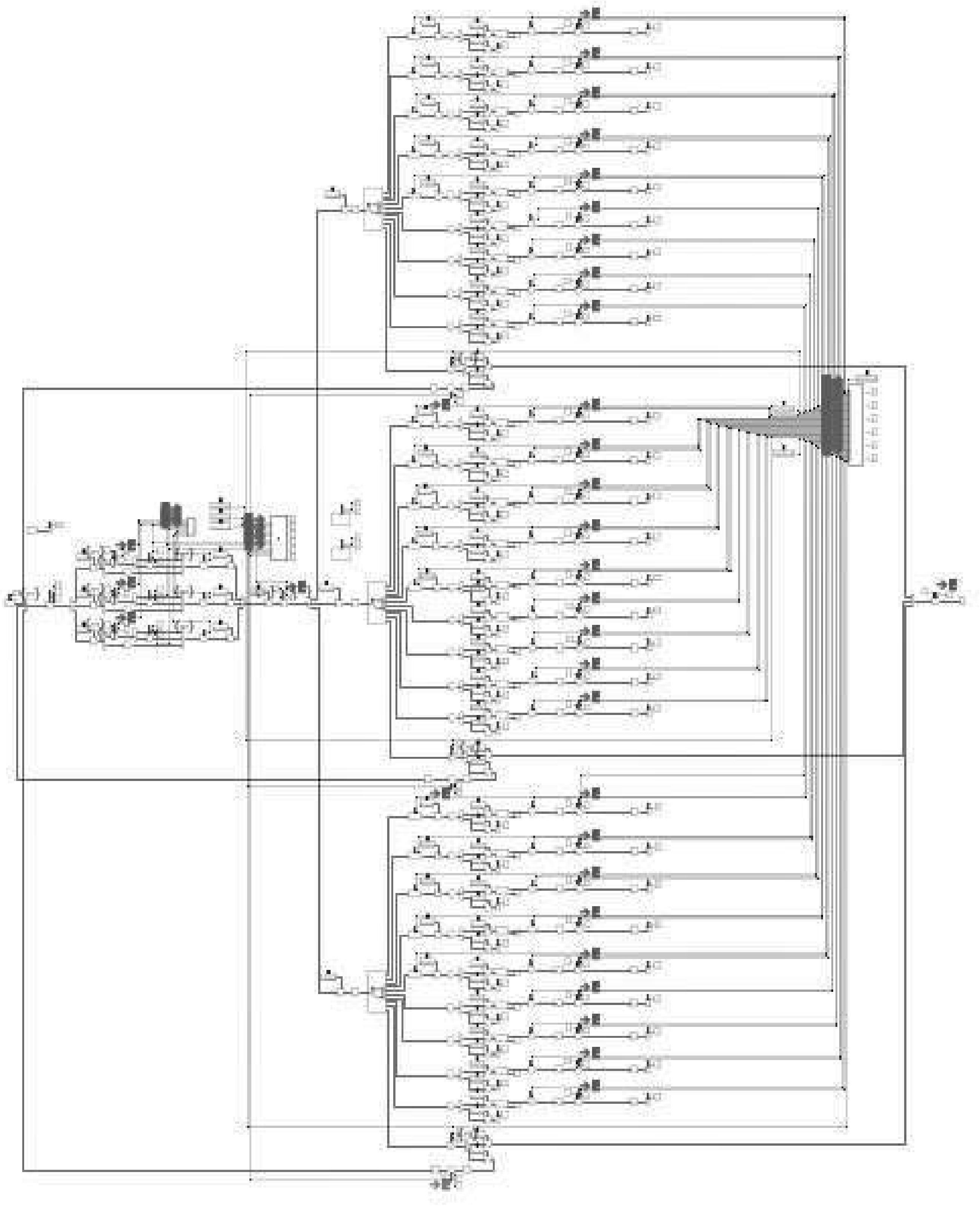


Figure 27 - System-wide diagram

4.4. Model calibration parameters.

Each server block used in the model requires a set of service time values for its application, which means that the correct choice of these times ends up being the way the entire model is calibrated. It is therefore very important to describe the different time parameters used and their origin, as explained below:

For the server cranes time we have the following logical sequence:

1. The crane is positioned on containers: I use the values from step 5.
2. Descends 6 meters (empty): (6 [m]) / (12 [m/min]).
3. Take the container: 0.3 minutes.
4. Climbs 6 meters loaded: (6 [m]) / (9 [m/min]).
5. Scroll to the objective:
    a. Train to Train: (6.13 [m]) / (115 [m/min]).
    b. Train to Camion: (8.20 [m]) / (115 [m/min]).
    c. Train to Storage: (19 [m]) / (115 [m/min]).

    Where 6.13, 8.2 and 19 are average values. For example, between track 1 and truck are 12.8 m, between track 2 and truck are 8.2 m and between track 3 and trucks 3.6 m. The average is equal to 8.2 m.
6. Descends again: (6 [m]) / (9 [m/min]).
7. Download: 0.3 minutes.
8. Goes up (empty): (6 [m]) / (12 [m/min]).
9. Translation movement: (12.5 [m]) / (60 [m/min]).

    Here the distance can be 0 m if, for example, you are working with Train 1 on Trucks and with Train 2 on Train 3. Also, it can be 6.2 m which is the distance between TEU and TEU. I chose 12.5 as an average value because it may be that the crane must transfer more than 2 TEU to work.
10. Begin again.

All the speed values used correspond to the ones mentioned in a deliverable of a research project in the logistic field [26].

- Load lifting speed: 9-12 m/min.
- Vacuum lifting speed: 12-18 m/min.
- Scrolling speed: 100-130 m/min.
- Side shift speed: 20-70 m/min.

Finally, we have:

Kind of interchange	**Mean [min]**	**Variance [min^2]**
Train to Train	1.81463	0.000564062
Train to Track	1.85333	0.002081549
Train to Storage	1.94333	0.002559348

Track to Train	1.85333	0.002081549
Storage to Train	1.94333	0.002559348

Table 2 - Server cranes time

The other times are discussed below:

	[Min]	
ITUs and wagons check-in:	**30**	Elaboration on literature data [27]
*changing locomotives		
*Load Verification Operation		
*Document check		
Service Time Broken Wagons:	**5 to 20**	Elaboration on literature data.
*manoeuvring time to bring a breakdown wagon to a depot		
ITUs and wagons check-out	**5**	Estimated value
Return	**5**	Estimated value
*arrive locomotive		
*arrive at the staging tracks		
Train crossing	**3.12**	Speed of 15 km/h and 750 m of the convoy + 30 m of the exchanger
*manoeuvring time to reach the operational track		

Table 3 - Other times

4.5. Output parameters.

The general output of Simulink and SimEvents is the Scope block, which gives a graphical format to the data output, having in abscissa the time values from the beginning of the simulation to the end, and in order the values of the signal being plotted. It is an elegant way to see the evolution of a certain internal parameter of the model during the simulation time.

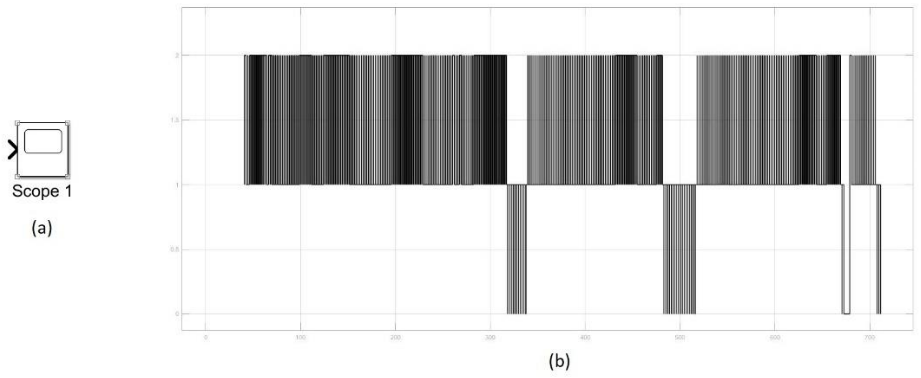
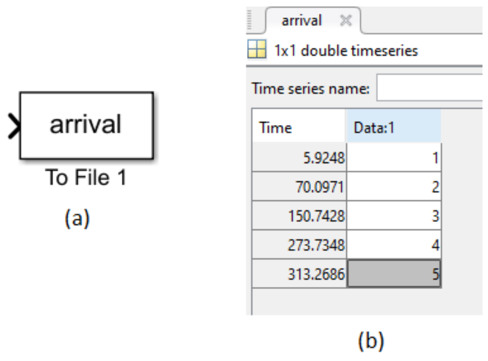


Figure 28 - (a) Scope block - (b) Example of output of Scope block.

The Scope block has the disadvantage of not allowing direct numerical data collection, so for a more detailed analysis, it is necessary to incorporate another block called "To File" in order to obtain the desired values in table form. It should be noted that this table is viewed from the central body of Matlab and not from SimEvents.



Time	Data:1
5.9248	1
70.0971	2
150.7428	3
273.7348	4
313.2686	5

Figure 29 - (a) To File block - (b) Example of table of a To File block.

Each type of block within SimEvents can generate certain types of output data called statistics, these can be graphically viewed by the already named Scope block.

For the ***Server block*** these statistics are:

- *Number of entities departed, d.* It allows us to know the number of entities that have been served and have left the block. This data is useful to obtain the time values in which the trains have departed from the station.
- *Number of entities in block, n*. Allows to know the number of entities being served inside the block. This is useful to know if, when crossing trains from pick-up track to operational track, two trains are compatible.
- *Pending entity in block, pe*. indicates whether a queue of entities waiting to be served has been generated. In the way the model is presented, this value should always be zero, giving us a good parameter when designing.
- *Number of pending entities, np*. In addition to "Pending entity in block", it indicates the number of entities that are waiting to join the block.
- *Average wait, w.* Complete the two previous items by returning an average waiting time.
- *Utilization, util.* Indicates whether the block is in operation or not. It is very useful for creating conditions when it comes to allowing trains to pass and move inside the station.

For the ***Entity Generator block*** are:

- *Number of entities departed, d*. It allows us to know the number of entities that have been generated and have left the block. This data is useful to obtain the time of arrival of the trains at the station.
- *Pending entity present in block, pe.* Indicates if there are entities that for some reason cannot advance in the model.
- *Average intergeneration time, w.* Returns an average time value between the generation of one entity and the next.

For the *Resource Pool block* they are:

- *Amount in use, or*. Indicates the amount of the own resource of the block being used. Very useful for obtaining information on the use of cranes and locomotives.
- *Average utilization, util.* Indicates the average time of use of the resource.
- *Amount available, avail.* Against "Amount in use" indicates the amount of the resource we have available for use.

For the *Resource Acquire block* are:

- *Number of entities departed, d.* It allows us to know the number of entities that have acquired a certain resource and have left the block.
- *Number of entities in block, n.* Allows to know the number of entities that are waiting inside the block to acquire a given resource.
- *Average wait, w.* Provides information on the above item.

For the *Entity Queue block* they are:

- *Number of entities departed, d.* It allows us to know the number of entities that have passed through the block.
- *Number of entities in block, n.* It lets you know how many entities are in the queue waiting for the path to continue.
- *Average wait, w.* Indicates the average value of the entities within the block.
- *Average queue length, l.* Indicates the average value of entities that are in the queue waiting for the path to be released to continue.

For the Entity Terminator block it is only "Number of entities arrived, a", which indicates the number of entities entering the block.

Also, the Matlab function block can generate output data, but through functions within the same block. For example, we can create an output that is the sum of several signals that enter the block and, through the Scope block, see graphically the evolution of this sum over time.

5. Scenarios

The main objective of this section is to propose and describe enough scenarios so that, when simulating them, it can be determined by the results obtained in each of them, if the model designed is functional or not. It will also be useful to determine which phenomena associated to the variations of the parameters in the different scenarios can be studied by means of the proposed simulation. For the model to be functional, logical results must be obtained, for example by adding more cranes it is expected that the capacity of the station will be increased or, as a limit, maintained, but not reduced. For this I have chosen 11 different scenarios, where the parameters number of locomotives, number of cranes, arrival train delays, number of failure wagons, infrastructure and the origin-destination of ITUs will vary.

1. A first normal scenario, to be compared with the others, which I will simply call "Baseline scenario" and which has 2 locomotives, 2 cranes, common plant, minor delay, few failure wagons and an origin/destination of the ITUs type X/X/X between train, track and storage.
2. Baseline scenario with one extra crane.
3. Baseline scenario with two extra cranes.
4. Baseline scenario with two extra shunting locomotives.
5. Baseline scenario with improved infrastructure.
6. Baseline scenario with a 100% train-to-train O/D changeover.
7. Baseline scenario with a 100% train-truck O/D changeover.
8. Baseline scenario with a 50% train-to-train, 50% train-to-truck O/D changeover.
9. Baseline scenario with only 1 locomotive, 1 crane, long delay and many breakdown wagons. Called "Poor scenario".
10. Baseline scenario with many locomotives, 4 cranes, no delay, no breakdown wagons and improved infrastructure. Called "Ideal scenario".
11. Baseline scenario with delayed arrival trains.

With scenarios 1,2,3,4,5 we can see the influence of any improvement. With scenarios 1,6,7,8 we can see the influence of the origin-destination type of UTIs. With scenario 9 we get an idea of the worst that can be expected, without completely stopping the station. And, on the contrary, the number 10 is useful to know what the best thing is to aspire to. The scenario 11 tells us whether the system increases, decreases or maintains the delays.

All these changes are easily made in the simulation. Just change the corresponding values and the appropriate command lines:

- For scenarios 2,3 and 4 it is only necessary change the corresponding value of the Resources Pool.
- For scenario 5 we must change the values of the matrix of value for trains crossing the following ones:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Matrix 2 - Scenario 5: improved infrastructure.

- For scenarios 6, 7 and 8 we must change the values of the matrix Origin/Destination:

$$\begin{bmatrix} 0 & 26 & 27 & 25 & 32 & 0 & 0 \\ 28 & 0 & 28 & 27 & 27 & 0 & 0 \\ 26 & 31 & 0 & 28 & 27 & 0 & 0 \\ 32 & 23 & 28 & 0 & 25 & 0 & 0 \\ 21 & 29 & 28 & 29 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Matrix 3 - Scenario 6: 100% Train-Train

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 110 & 0 \\ 0 & 0 & 0 & 0 & 0 & 110 & 0 \\ 0 & 0 & 0 & 0 & 0 & 110 & 0 \\ 0 & 0 & 0 & 0 & 0 & 110 & 0 \\ 0 & 0 & 0 & 0 & 0 & 110 & 0 \\ 110 & 110 & 110 & 110 & 110 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Matrix 4 - Scenario 7: 100% Train-Truck

$$\begin{bmatrix} 0 & 13 & 14 & 10 & 19 & 54 & 0 \\ 15 & 0 & 15 & 11 & 14 & 55 & 0 \\ 12 & 17 & 0 & 14 & 13 & 54 & 0 \\ 19 & 10 & 15 & 0 & 12 & 54 & 0 \\ 8 & 16 & 15 & 16 & 0 & 55 & 0 \\ 56 & 54 & 51 & 59 & 52 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Matrix 5 - Scenario 8: 50% Train-Train, 50% Train-Truck

- In scenario 7, moreover, we must simplify many conditions, because in this case, trains can be served without a second cycle. For example, in "PRINCIPAL GATE CONTROL 2", we modify the opening conditions of the Gates, so that only the exit gates are opened and not the return gates.
- In scenario 9, it must be borne in mind that there will be only one locomotive, so we must avoid it being blocked at any point. For the train delays we must make change in the “Entity Generator” block. The same for the quantity of broken Wagons, but in the “Entity Server” correct block. About the locomotive we have changed on the main train track number 3, the blocks of "Resource Acquire" and "Resource Release".

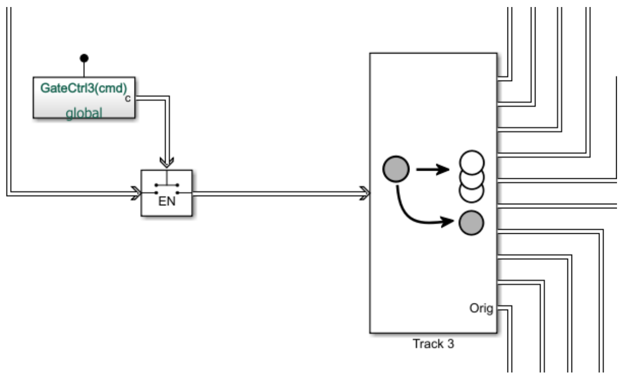


Figure 30 - New track system for scenario 9

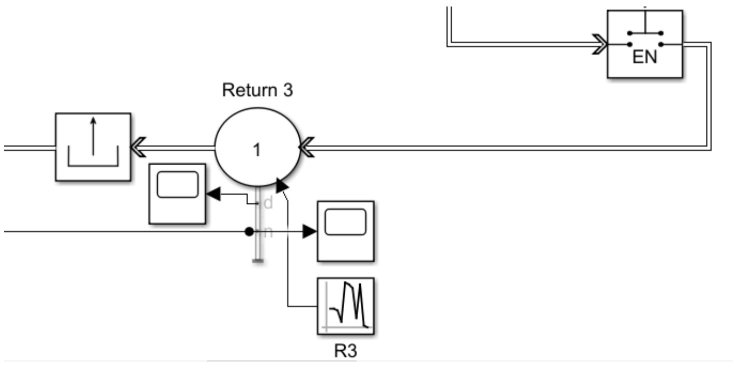


Figure 31 - New return system for scenario 9

- On the scenario 10, several of the above-mentioned changes have been made: the changes in scenarios 3, 4 and 5, plus some changes like those in scenario 9, but as improvements.
- In the last scenario, all you must do is change the generation values in the "Entity Generator" block.

To obtain statistically significant results, each scenario was simulated a number equal to 30 times, a value obtained from approximating with confidence 99% the right tail of probability 0.05 of an unknown distribution, with an imprecision level of 0.1 minutes.

6. Results

After making the corresponding adjustments for each scenario, each one was simulated separately. Next, we will see a series of tables with the time values obtained for each of the simulations, the comparisons made between them and various calculations.

In the first stage, the entry and exit times of each train, the time of use of the locomotives and the time of use of the cranes for each scenario were recorded for the scenarios 1 to 11:

Scenario	Arrival					Departure					Locomotive Use		Cranes Use	
	Train 1	Train2	Train 3	Train 4	Train 5	Train 1	Train2	Train 3	Train 4	Train 5	From	To	From	To
	[min]					[min]					[min]	[min]	[min]	[min]
1	1.0	16.0	26.0	31.0	56.0	532.6	556.7	561.7	593.2	727.7	1.0	689.9	51.6	722.1
2	1.0	16.0	26.0	31.0	56.0	388.3	399.1	416.9	448.0	544.6	1.0	507.2	53.4	539.8
3	1.0	16.0	26.0	31.0	56.0	318.3	342.7	347.5	378.6	452.5	1.0	414.6	47.0	447.3
4	1.0	16.0	26.0	31.0	56.0	532.5	557.7	562.5	595.9	729.0	1.0	691.2	50.7	723.8
5	1.0	16.0	26.0	31.0	56.0	527.1	550.9	555.6	569.9	721.5	1.0	684.1	47.4	716.7
6	1.0	16.0	26.0	31.0	56.0	394.5	440.0	444.9	494.9	603.9	1.0	547.7	59.3	598.6
7	1.0	16.0	26.0	31.0	56.0	464.3	668.0	672.7	872.6	1087.1	1.0	877.4	51.2	1081.6
8	1.0	16.0	26.0	31.0	56.0	624.8	643.4	647.9	672.8	843.2	1.0	811.1	51.2	838.3
9	14.7	61.8	99.5	143.9	197.6	1050.4	1081.9	1331.5	1403.2	1463.1	14.7	1337.8	72.2	1458.0
10	1.0	1.0	1.0	1.0	1.0	306.9	321.0	335.6	367.1	440.6	1.0	402.7	34.2	435.4
11	55.9	132.9	199.4	275.9	358.2	589.0	600.1	632.6	661.1	792.9	55.9	616.3	100.8	787.6

Table 4 - First Results

The purpose of scenario 11 is to establish the response of the model to delays. Various cases were simulated, varying the times of delays and the trains that were delayed. The result is that the system generally recovers delays if they occur starting from the second train and in values maximum from 100 minutes, for the second train, to 420 minutes, for the last train. For higher values, delays begin to occur but always of lower value than the incoming delay. In the case of an overall delay, this only moves, does not increase or decrease, in other words, if everything starts running with a delay of two hours, the trains will leave, on average, with a delay of two hours.

This second table analyses the number of hours the system was in operation. The first part represents the total time from the time the first train entered until the last train withdraws. The second is the use of locomotives. And the third, the use of cranes. Finally, comparisons were made between the various scenarios in front of scenario 1, where the values where the time is less have been coloured green, the values where the time is less, the values where the time is equal are black and the values where the time is greater are red:

Scenario	System Hours			Locomotive Use			Cranes Use		
	From	To	Total	From	To	Total	From	To	Total
	[min]	[min]	[hours]	[min]	[min]	[hours]	[min]	[min]	[hours]
1	1.0	727.7	**12.11**	1.0	689.9	**11.48**	51.6	722.1	**11.18**
2	1.0	544.6	**9.06**	1.0	507.2	**8.44**	53.4	539.8	**8.11**
Compared to S1	*0.0*	*183.1*	*3.05*	*0.0*	*182.7*	*3.05*	*-1.9*	*182.2*	*3.07*
3	1.0	452.5	**7.53**	1.0	414.6	**6.89**	47.0	447.3	**6.67**
Compared to S1	0.0	275.2	**4.59**	0.0	275.3	**4.59**	4.6	274.7	**4.50**
4	1.0	729.0	**12.13**	1.0	691.2	**11.50**	50.7	723.8	**11.22**
Compared to S1	0.0	-1.3	**-0.02**	0.0	-1.3	**-0.02**	0.8	-1.8	**-0.04**
5	1.0	721.5	**12.01**	1.0	684.1	**11.38**	47.4	716.7	**11.16**
Compared to S1	0.0	6.2	**0.10**	0.0	5.8	**0.10**	4.2	5.3	**0.02**
6	1.0	603.9	**10.05**	1.0	547.7	**9.11**	59.3	598.6	**8.99**
Compared to S1	0.0	123.8	**2.06**	0.0	142.2	**2.37**	-7.7	123.5	**2.19**
7	1.0	1087.1	**18.10**	1.0	877.4	**14.61**	51.2	1081.6	**17.17**
Compared to S1	0.0	-359.4	**-5.99**	0.0	-187.5	**-3.12**	0.3	-359.6	**-6.00**
8	1.0	843.2	**14.04**	1.0	811.1	**13.50**	51.2	838.3	**13.12**
Compared to S1	0.0	-115.5	**-1.92**	0.0	-121.2	**-2.02**	0.3	-116.3	**-1.94**
9	14.7	1463.1	**24.14**	14.7	1337.8	**22.05**	72.2	1458.0	**23.10**
Compared to S1	-13.7	-735.4	**-12.03**	-13.7	-647.9	**-10.57**	-20.6	-736.0	**-11.92**
10	1.0	440.6	**7.33**	1.0	402.7	**6.70**	34.2	435.4	**6.69**
Compared to S1	0.0	287.1	**4.78**	0.0	287.2	**4.79**	17.3	286.6	**4.49**
11	55.9	792.9	**12.28**	55.9	616.3	**9.34**	100.8	787.6	**11.45**
Compared to S1	-54.9	-65.2	**-0.17**	-54.9	73.6	**2.14**	-49.3	-65.6	**-0.27**

Table 5 - Scenarios comparison

This comparison yields values that were expected from the beginning:

- The addition of more cranes significantly reduces the time required.
- The addition of locomotives does not makes a significant difference. The new locomotives are not almost used due to the low train traffic (Figure 29).
- Something similar happens with scenario 5, for the same reasons as before, there are few trains at separate times, so there are no meetings. Surely in a simulation with many trains arriving simultaneously this improvement would have produced a decrease in time.
- Scenario 6 shows a decrease of 2.19 hours, which leads us to believe that it is better to work only with exchanges between trains. This is relative, as we will see later that this is not the case.
- Similar but opposite situations are presented in scenarios 7 and 8.

- In the worst simulated condition, without completely stopping the operation of the system, it gives us an increase of 11.92 hours, almost double the time.
- On the other hand, scenario 10 with 4.49 hours gives a high favourable value, as expected.

We can affirm, by means of these results, that the chosen model has a functioning in agreement with the expected thing for the type of simulation, therefore, it is a useful model.

In addition to the results mentioned above, we can add that the use of "Scope" blocks is useful for analysing the use of cranes and locomotives. We can establish a ratio "U" between the total available area, defined as the product of the number of cranes or locomotives available and the total time they are used, and the net area, defined between the initial and final values of the time of use and between the line of the abscissa and the function drawn, that is to say the area under the curve. This ratio could be used as an indicator of the relative use of different machines. In the following figures we will see the use of these for the first scenarios.

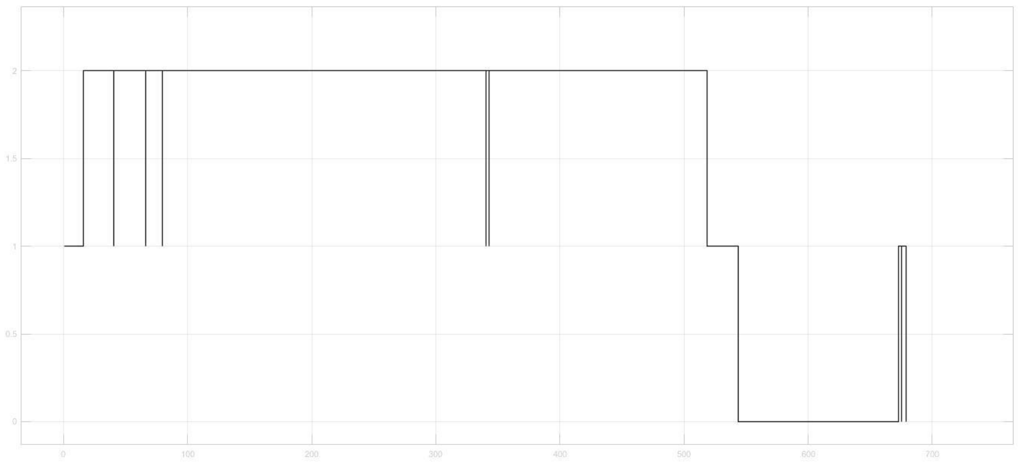


Figure 32 - Use of locomotives on Scenario 1. U = 0.78.

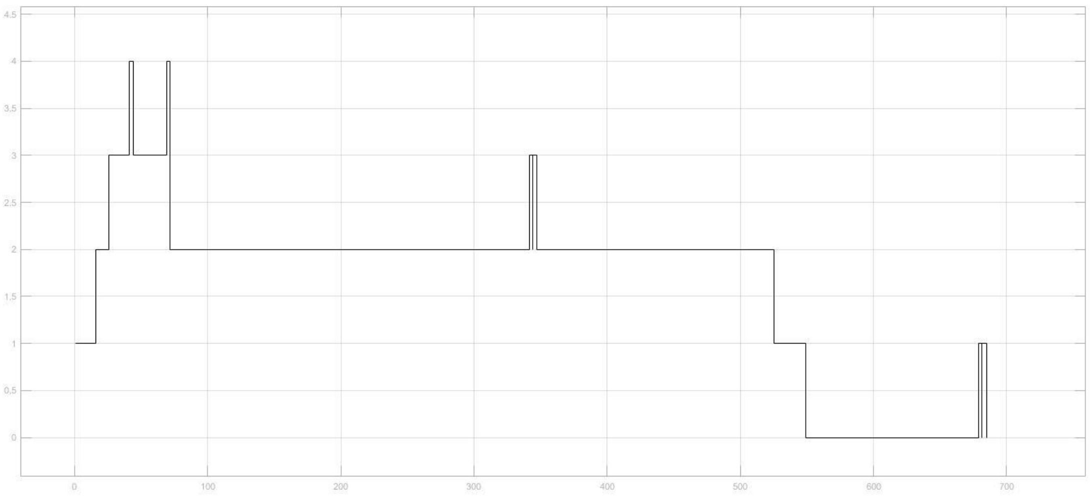


Figure 33 - Use of locomotives on Scenario 4. U = 0.41.

We can see that two locomotives are quite used in most of the process, but when we decided to add two more, they end up being used only at the beginning, concluding that this is not efficient.

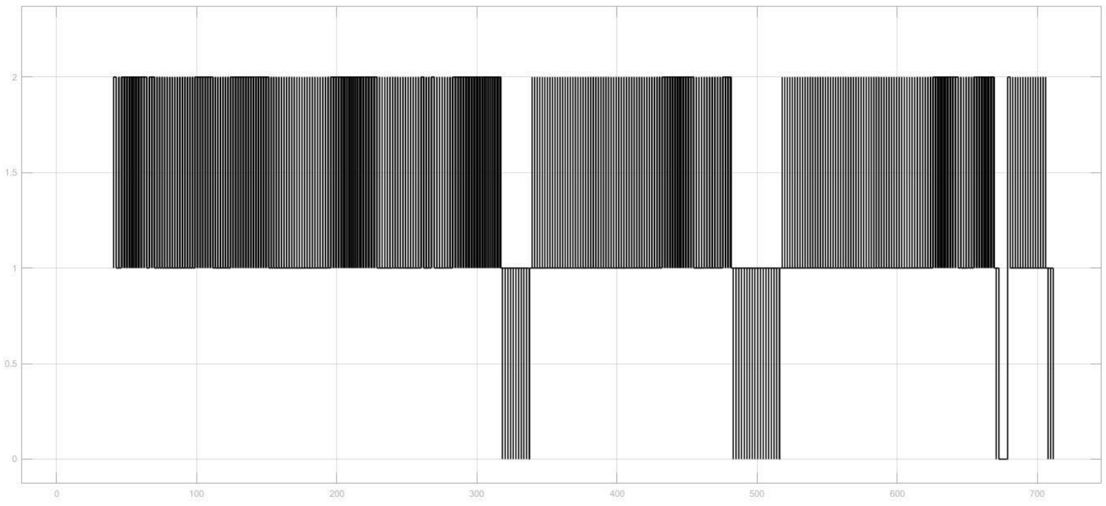


Figure 34 - Use of Cranes on Scenario 1. U = 0.94.

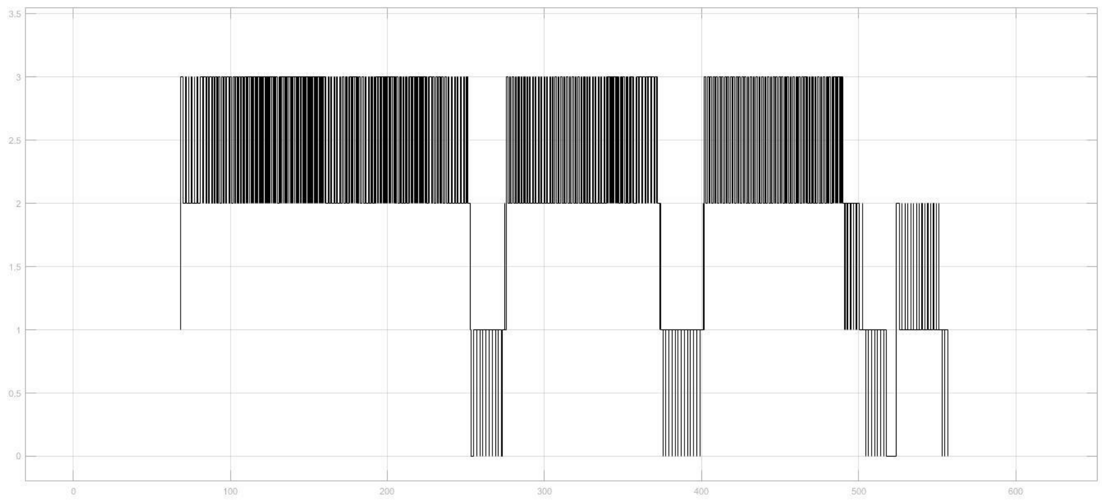


Figure 35 - Use of Cranes on Scenario 2. U = 0.86.

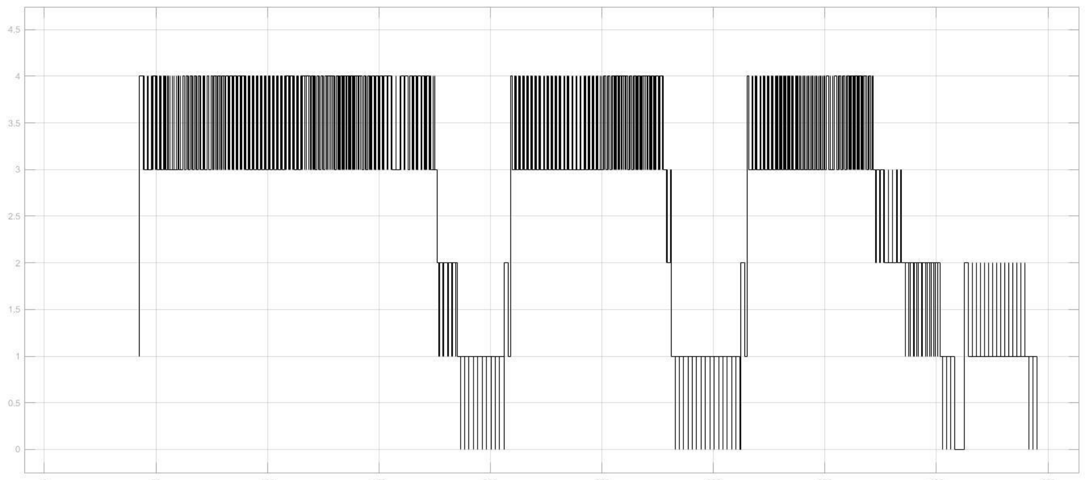


Figure 36 - Use of Cranes on Scenario 3. U = 0.79.

On the other hand, in these figures we can see that each crane added is very well used.

Being a very versatile simulation model, you can study many other questions such as:

- influence of track lengths within the station,
- planning train arrival times,
- implementation of different internal manoeuvres,
- It is possible to modify the prioritization, to serve trucks first, or to serve the storage area.

Also, we can use the results to obtain the Key Performance Indicators (KPIs) [28]. In the follow table, I propose the analysis with a KPI which is the "Equipment Performance". It is the potentiality of handling equipment:

$$E_p = \frac{n\ ITU}{h}$$

Where:

- n ITU = number of handled intermodal transport unit.
- h = hour.

The number of ITUs is obtained by the sum of all elements of the origin/destination matrices.

Scenario	Cranes Use			Total	Equipment
	From	To	Total	ITUs	Performance
	[min]	[min]	[hours]	Served	[ITU/hour]
1	51.6	722.1	**11.18**	682	**61.03**
2	53.4	539.8	**8.11**	682	**84.13**
Compared to S1	*-1.9*	*182.2*	*3.07*		*23.10*
3	47.0	447.3	**6.67**	682	**102.22**
Compared to S1	4.6	274.7	**4.50**		41.19
4	50.7	723.8	**11.22**	682	**60.79**
Compared to S1	0.8	-1.8	**-0.04**		-0.24
5	47.4	716.7	**11.16**	682	**61.13**
Compared to S1	4.2	5.3	**0.02**		0.10
6	59.3	598.6	**8.99**	547	**60.85**
Compared to S1	-7.7	123.5	**2.19**		-0.18
7	51.2	1081.6	**17.17**	1100	**64.06**
Compared to S1	0.3	-359.6	**-6.00**		3.03
8	51.2	838.3	**13.12**	822	**62.66**
Compared to S1	0.3	-116.3	**-1.94**		1.63
9	72.2	1458.0	**23.10**	682	**29.53**
Compared to S1	-20.6	-736.0	**-11.92**		-31.50
10	34.2	435.4	**6.69**	682	**101.99**
Compared to S1	17.3	286.6	**4.49**		40.96
11	100.8	787.6	**11.45**	682	**59.58**
Compared to S1	-49.3	-65.6	**-0.27**		-1.45

Table 6 - Scenarios comparison 3

For scenarios 2, 3, 4 and 5, the above statements are repeated: adding cranes produces significant improvements but adding locomotives and improving the tracks not so much.

The most interesting thing to see is that scenario 6, despite being faster, so one would think that it is an improvement, the potential is less, since fewer ITUs are served. This is because to be able to serve all the trains, it is necessary to carry out recoil manoeuvres where time is wasted. In scenario 7, however, these manoeuvres are not necessary as each train can be operated independently of the other trains. The difference in the latter scenario is almost 3.03 ITUs per hour served.

Similar conclusions can be reached with scenario 8, the times are higher, but the number of ITUs is also higher, there are no exchanges between trains and warehouses that are the slowest and the potential increases by almost 2 units.

After all the analyses, we reiterate the confirmation that the model chosen and simulated is adequate and coherent with reality. No anomalous values have been observed to question the model.

7. Conclusion

At the beginning of this work we can see that SimEvents has not been specifically used before for the creation of a model of an intermodal station which makes this thesis valuable in providing a new model and point of view. On the contrary, this part also shows that the software has been used in applications related to the world of intermodal transport and that the simulation of intermodal stations is widely seen.

Following this, the various hypotheses and design conditions that have been chosen to have been listed, which facilitate the design of the model to be studied without necessarily for the general research. As we know, the study was based on the creation of a simulation model that allows, more or less broadly, to generate an intermodal station with all its sub phenomena and simulate various scenarios that would be feasible to happen in reality, ranging from the choice of improvements, through the change of configurations of origin / destination of loads to the simulation of unfavourable cases. All of this allows key output data to be obtained for the operation of these terminals.

After detailing how each part of the model was carried out, avoiding multiple setbacks and always with the objective of making it faithful to reality, a stable schedule was arrived at which, in view of the large number of times that the different scenarios have been simulated, always obtaining logical results and at the same time ensuring the fidelity of the results obtained, it is useful for logistics and for the planning of new and existing intermodal terminals. It is evident that the model allows the study of many other parameters that have been left out of the ones illustrated in this paper, showing that the model has multiple potential applications. In addition, this programme can be supplemented by various cost-benefit analyses, for example, to obtain ideal scenarios for activity within the intermodal station.

The simulation of the scenarios corroborated the expected results, confirming the usefulness of the model, for example:

- In the case of small terminals, the increase in the number of cranes by one unit or more means significant performance improvements. Not being the same for the increase of locomotives or for the improvement of facilities.
- The type of exchange that maximizes performance is the 100% Train-Truck type, since no time is wasted on the extra manoeuvres that the trains must carry out in other cases.
- In the most unfavourable case, it is not enough for an entire day (24.67 hours) to finish the train service, which would influence the activities of the following day.
- The ideal case is rather an anecdotal one, but when comparing the result with the one obtained in scenario 3, we come to the same conclusion as before: the increase in the number of cranes is the best improvement proposed, since the others do not have as much weight.
- The different arrival and departure values of the trains of the last scenario allow us to see that the model created is influenced by such values, seeing that the

variations between the arrivals and their corresponding departures are not maintained during the process.

I believe that, despite not having a very high level of programming, all the objectives set out in the introduction have been met satisfactorily, which gives me immense joy and a feeling of satisfaction at knowing that all the hours spent on this project have been fruitful. I am also pleased with the clarity with which the project has been detailed, being confident that anyone interested in this topic can replicate this work for their benefit or to continue their research.

Despite this satisfaction I am aware that the topic can be broadened so, based on the experience developed during this whole study, I can propose several suggestions for future work. These are the ones:

- To optimize the program through: a more exhaustive investigation of the times used, and the comparison of the simulated results with real results.
- The system could be further automated, for example by avoiding the implementation of destination conditions for each train.
- Although it is possible to add/remove several trains and tracks, it is arduous work. Consider how to expedite this.
- Another very interesting point would be the creation of a subsystem that considers the movement and generation of queues in trucks, which interacts with what has already been done.
- Perform updates with the new tools that may be available in new versions of the software.
- From a purely computer point of view, a Matlab®-independent application can be created, with the corresponding entry of the desired characteristic values.

I have come to the conclusion that the understanding of the different software present in the field of civil engineering (and engineering in general), together with the knowledge of programming languages and the use of general software, such as Matlab, is very important for the current development of our profession. In my previous studies I have not had such deep contact with these design and programming tools, which I deeply regret. Fortunately, the time spent in this institution has allowed me to have this longed-for contact, generating in me a greater curiosity about these aspects. Needless to say, the application of these tools should never be done alone, but in the company of professional experience outside of computing.

References

[1] UIRR Report, EUROPEAN ROAD-RAIL COMBINED TRANSPORT, 2017-18.
[2] SCHINDLBACHER E., HÄUSLMAYER H., GRONALT M., “Deliverable 3 Modelling and simulation of intermodal terminal networks”, 2011.
[3] TROCHE G., “EvaRail – activity-based transport cost model for evaluation of improvements in the rail freight system”.
[4] GRONALT M., BENNA T., POSSET M., SimConT Simulation of hinterland Container Terminal operations, 2015.
[5] [6] MINT – model and decision support systems for intermodal terminal networks, Introduction, 2011.
[7] DALLA CHIARA B., MANTI E., MARINO M., “Intermodal terminals with gateway function: simulation of their engineering on a case study”, CIFY 2013.
[8] http://www.automod.it.
[9] http://vision-traffic.ptvgroup.com.
[10] VISSIM – State-of-the-Art Multi-Modal Simulation-
[11] CAMAJ J., MASEK J., KENDRA M., “Simulation of the Transport Centre Process by Using Special Tools”, 2015.
[12] KARAKOSTAS B., “Cooperative intermodal operations using a publish-subscribe approach”, 2018.
[13] GALONSKE N., RIEBE E., TOUBOL A. WEISMANTEL S., “The ViWaS project: future-proof solutions for wagonload transport”, 2016.
[14] CHEN X., HE S., LI T., LI Y., “A Simulation Platform for Combined Rail/Road Transport in Multiyards Intermodal Terminals”, 2018.
[15] VELOQUI M., TURIAS I., CERBAN M. M., GONZALEZ M. J., BUIZA G. BELTRAN J., “Simulating the landside congestion in a container terminal. The experience of the port of Naples (Italy)”, 2014.
[16] RUIZ-AGUILAR J. J., TURIAS I. J., CERBAN M., JIMENEZ-COME M. J., PULIDO A., “Time analysis of the containerized cargo flow in the logistic chain using simulation tools: the case of the Port of Seville (Spain)”, 2016.
[17] RAIA P., “Calcolo della capacità di nodi ferroviari: un metodo di simulazione a eventi discreti”, 2015.
[18] JIMENEZ D. C., “Simulación de terminals ferroviarias de transporte intermodal con zona de clasificación”, 2015.
[19] GHAFFARI KHAN O. A., “Analysis and Scheduling of machinery in an Intermodal Terminal by using the OSPF concept”, 2008.
[20] ZHAO Y., LIU R., ZHANG X., WHITEING A., “A chance-constrained stochastic approach to intermodal container routing problems”, 2018.
[21] https://en.wikipedia.org/wiki/MATLAB#Release_history.
[22] https://en.wikipedia.org/wiki/List_of_discrete_event_simulation_software.
[23] https://it.mathworks.com/products/simevents/model-examples.html.
[24] DALLA CHIARA B., MARIGO D., BENZO G., “Interporti e terminali intermodali”, HOEPLI 2002.
[25] SimEvents®: Getting Started Guide R2017b – MathWorks, 2017.
[26] Notes of “Logistica e Trasporti” Prof. RIZZO, CMN CMA.
[27] DALLA CHIARA B., *"Sistema di trasporto intermodali: progettazione ed esercizio"*, EGAF 2009.
[28] RICCI S., CAPODILUPO L., MUELLER B., KARL J., SCHNEBERGER J., “Assessment methods for innovative operational measures and technologies for intermodal freight terminals”, 2016.

Appendix

Command lines for PRINCIPAL GATE CONTROL 1

```
function [DestA,DestB,DestC,x,y,z] =
fcn(DestinationA,DestinationB,DestinationC,P,X,Y,Z,g1,g2,g3,r1,r2,r3)

open = 1;
close = 0;

DEST = [ X Y Z ];

M1 = [ 0 0 0  0 1 1  0 1 1 ];
M2 = [ 0 0 0  0 0 1  0 0 1 ];
M3 = [ 0 0 0  0 0 0  0 0 0 ];

M4 = [ 0 0 0  0 0 0  0 1 1 ];
M5 = [ 1 0 0  0 0 0  0 0 1 ];
M6 = [ 1 1 0  0 0 0  0 0 0 ];

M7 = [ 0 0 0  0 0 0  0 0 0 ];
M8 = [ 1 0 0  1 0 0  0 0 0 ];
M9 = [ 1 1 0  1 1 0  0 0 0 ];


M = [M1;M2;M3;M4;M5;M6;M7;M8;M9];
if P == 0
    if (DestinationA == 0) && (DestinationB == 0) && (DestinationC == 0)
        GateCtrlA1(close);
        GateCtrlB1(close);
        GateCtrlC1(close);
        GateCtrlP(close);
    elseif (DestinationA == 0) && (DestinationB == 0) && (DestinationC ~= 0)
        if DEST(DestinationC) == 0
            GateCtrlA1(close);
            GateCtrlB1(close);
            GateCtrlC1(open);
            GateCtrlP(open)
        else
            GateCtrlA1(close);
            GateCtrlB1(close);
            GateCtrlC1(close);
            GateCtrlP(close);
        end
    elseif(DestinationA == 0) && (DestinationB ~= 0) && (DestinationC == 0)
        if DEST(DestinationB) == 0
            GateCtrlA1(close);
            GateCtrlB1(open);
            GateCtrlC1(close);
            GateCtrlP(open);
        else
            GateCtrlA1(close);
            GateCtrlB1(close);
            GateCtrlC1(close);
            GateCtrlP(close);
        end
    elseif(DestinationA ~= 0) && (DestinationB == 0) && (DestinationC == 0)
        if DEST(DestinationA) == 0
            GateCtrlA1(open);
            GateCtrlB1(close);
            GateCtrlC1(close);
            GateCtrlP(open);
        else
            GateCtrlA1(close);
            GateCtrlB1(close);
            GateCtrlC1(close);
            GateCtrlP(close);
        end
    elseif(DestinationA ~= 0) && (DestinationB ~= 0) && (DestinationC == 0)
        if M(DestinationA,(3 + DestinationB)) == 0
            if (DEST(DestinationA) == 0) && (DEST(DestinationB) ~= 0)
                GateCtrlA1(open);
                GateCtrlB1(close);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) == 0)
                GateCtrlA1(close);
                GateCtrlB1(open);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) ~= 0)
                GateCtrlA1(close);
                GateCtrlB1(close);
                GateCtrlC1(close);
                GateCtrlP(close);
            elseif (DEST(DestinationA) == 0) && (DEST(DestinationB) == 0)
                GateCtrlA1(open);
    %Here I can decide which train goes by, B or A.
                GateCtrlB1(close);
                GateCtrlC1(close);
                GateCtrlP(open);
            end
        elseif M(DestinationA,(3 + DestinationB)) == 1
            if (DEST(DestinationA) == 0) && (DEST(DestinationB) ~= 0)
                GateCtrlA1(open);
                GateCtrlB1(close);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) == 0)
                GateCtrlA1(close);
                GateCtrlB1(open);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) ~= 0)
                GateCtrlA1(close);
                GateCtrlB1(close);
                GateCtrlC1(close);
                GateCtrlP(close);
            elseif (DEST(DestinationA) == 0) && (DEST(DestinationB) == 0)
                GateCtrlA1(open);
                GateCtrlB1(open);
                GateCtrlC1(close);
                GateCtrlP(open);
            end
        end
    elseif(DestinationA ~= 0) && (DestinationB == 0) && (DestinationC ~= 0)
        if M(DestinationA,(6 + DestinationC)) == 0
            if (DEST(DestinationA) == 0) && (DEST(DestinationC) ~= 0)
                GateCtrlA1(open);
                GateCtrlB1(close);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationC) == 0)
                GateCtrlA1(close);
                GateCtrlB1(close);
                GateCtrlC1(open);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationC) ~= 0)
                GateCtrlA1(close);
                GateCtrlB1(close);
                GateCtrlC1(close);
                GateCtrlP(close);
            elseif (DEST(DestinationA) == 0) && (DEST(DestinationC) == 0)
                GateCtrlA1(open);
    % Here I can decide which train goes by, A or C.
                GateCtrlB1(close);
                GateCtrlC1(close);
                GateCtrlP(open);
            end
        elseif M(DestinationA,(6 + DestinationC)) == 1
            if (DEST(DestinationA) == 0) && (DEST(DestinationC) ~= 0)
                GateCtrlA1(open);
                GateCtrlB1(close);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationC) == 0)
                GateCtrlA1(close);
                GateCtrlB1(close);
                GateCtrlC1(open);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationC) ~= 0)
                GateCtrlA1(close);
                GateCtrlB1(close);
                GateCtrlC1(close);
                GateCtrlP(close);
            elseif (DEST(DestinationA) == 0) && (DEST(DestinationC) == 0)
                GateCtrlA1(open);
                GateCtrlB1(close);
                GateCtrlC1(open);
                GateCtrlP(open);
            end
        end
    elseif(DestinationA == 0) && (DestinationB ~= 0) && (DestinationC ~= 0)
        if M(3 + DestinationB,(6 + DestinationC)) == 0
            if (DEST(DestinationB) == 0) && (DEST(DestinationC) ~= 0)
                GateCtrlA1(close);
                GateCtrlB1(open);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationB) ~= 0) && (DEST(DestinationC) == 0)
                GateCtrlA1(close);
                GateCtrlB1(close);
                GateCtrlC1(open);
                GateCtrlP(open);
            elseif (DEST(DestinationB) ~= 0) && (DEST(DestinationC) ~= 0)
                GateCtrlA1(close);
                GateCtrlB1(close);
                GateCtrlC1(close);
                GateCtrlP(close);
            elseif (DEST(DestinationB) == 0) && (DEST(DestinationC) == 0)
                GateCtrlA1(close);
    %Here I can decide which train goes by, B or C.
                GateCtrlB1(open);
                GateCtrlC1(close);
                GateCtrlP(open);
            end
        elseif M(3 + DestinationB,(6 + DestinationC)) == 1
            if (DEST(DestinationB) == 0) && (DEST(DestinationC) ~= 0)
                GateCtrlA1(close);
                GateCtrlB1(open);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationB) ~= 0) && (DEST(DestinationC) == 0)
                GateCtrlA1(close);
                GateCtrlB1(close);
                GateCtrlC1(open);
                GateCtrlP(open);
            elseif (DEST(DestinationB) ~= 0) && (DEST(DestinationC) ~= 0)
                GateCtrlA1(close);
                GateCtrlB1(close);
                GateCtrlC1(close);
            elseif (DEST(DestinationB) == 0) && (DEST(DestinationC) == 0)
                GateCtrlA1(close);
                GateCtrlB1(open);
                GateCtrlC1(open);
                GateCtrlP(open);
            end
        end
    elseif(DestinationA ~= 0) && (DestinationB ~= 0) && (DestinationC ~= 0)
        if (M(DestinationA,(3 + DestinationB)) == 0) && (M(DestinationA,(6 +
DestinationC)) == 0) && (M(3 + DestinationB,(6 + DestinationC)) == 0)
            if (DEST(DestinationA) == 0) && (DEST(DestinationB) == 0)&&
(DEST(DestinationC) == 0)
                GateCtrlA1(close);
    %Here I can decide which train goes by, A, B or C.
                GateCtrlB1(open);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) == 0) && (DEST(DestinationB) == 0)&&
(DEST(DestinationC) ~= 0)
                GateCtrlA1(close);
    %Here I can decide which train goes by, B or A.
                GateCtrlB1(open);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) == 0) && (DEST(DestinationB) ~= 0)&&
(DEST(DestinationC) == 0)
                GateCtrlA1(open);
                GateCtrlB1(close);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) == 0)&&
(DEST(DestinationC) == 0)
                GateCtrlA1(close);
    %Here I can decide which train goes by, B or C.
                GateCtrlB1(open);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) == 0) && (DEST(DestinationB) ~= 0)&&
(DEST(DestinationC) ~= 0)
                GateCtrlA1(open);
                GateCtrlB1(close);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) == 0)&&
(DEST(DestinationC) ~= 0)
                GateCtrlA1(close);
                GateCtrlB1(open);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) ~= 0)&&
(DEST(DestinationC) == 0)
                GateCtrlA1(close);
                GateCtrlB1(close);
                GateCtrlC1(open);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) ~= 0)&&
(DEST(DestinationC) ~= 0)
                GateCtrlA1(close);
                GateCtrlB1(close);
                GateCtrlC1(close);
                GateCtrlP(close);
            end
        elseif (M(DestinationA,(3 + DestinationB)) == 0) && (M(DestinationA,(6
+ DestinationC)) == 0) && (M(3 + DestinationB,(6 + DestinationC)) == 1)
            if (DEST(DestinationA) == 0) && (DEST(DestinationB) == 0)&&
(DEST(DestinationC) == 0)
                GateCtrlA1(close);
                GateCtrlB1(open);
                GateCtrlC1(open);
                GateCtrlP(open);
            elseif (DEST(DestinationA) == 0) && (DEST(DestinationB) == 0)&&
(DEST(DestinationC) ~= 0)
                GateCtrlA1(close);
    %Here I can decide which train goes by, B or A.
                GateCtrlB1(open);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) == 0) && (DEST(DestinationB) ~= 0)&&
(DEST(DestinationC) == 0)
                GateCtrlA1(open);
    %Here I can decide which train goes by, A or C.
                GateCtrlB1(close);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) == 0)&&
(DEST(DestinationC) == 0)
                GateCtrlA1(close);
                GateCtrlB1(open);
                GateCtrlC1(open);
                GateCtrlP(open);
            elseif (DEST(DestinationA) == 0) && (DEST(DestinationB) ~= 0)&&
(DEST(DestinationC) ~= 0)
                GateCtrlA1(open);
                GateCtrlB1(close);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) == 0)&&
(DEST(DestinationC) ~= 0)
                GateCtrlA1(close);
                GateCtrlB1(open);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) ~= 0)&&
(DEST(DestinationC) == 0)
                GateCtrlA1(close);
                GateCtrlB1(close);
                GateCtrlC1(open);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) ~= 0)&&
(DEST(DestinationC) ~= 0)
                GateCtrlA1(close);
                GateCtrlB1(close);
                GateCtrlC1(close);
                GateCtrlP(close);
            end
        elseif (M(DestinationA,(3 + DestinationB)) == 0) && (M(DestinationA,(6
+ DestinationC)) == 1) && (M(3 + DestinationB,(6 + DestinationC)) == 0)
            if (DEST(DestinationA) == 0) && (DEST(DestinationB) == 0)&&
(DEST(DestinationC) == 0)
                GateCtrlA1(open);
                GateCtrlB1(close);
                GateCtrlC1(open);
                GateCtrlP(open);
            elseif (DEST(DestinationA) == 0) && (DEST(DestinationB) == 0)&&
(DEST(DestinationC) ~= 0)
                GateCtrlA1(close);
    %Here I can decide which train goes by, B or A.
                GateCtrlB1(open);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) == 0) && (DEST(DestinationB) ~= 0)&&
(DEST(DestinationC) == 0)
                GateCtrlA1(open);
                GateCtrlB1(close);
                GateCtrlC1(open);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) == 0)&&
(DEST(DestinationC) == 0)
                GateCtrlA1(close);
    %Here I can decide which train goes by, B or C.
                GateCtrlB1(open);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) == 0) && (DEST(DestinationB) ~= 0)&&
(DEST(DestinationC) ~= 0)
                GateCtrlA1(open);
                GateCtrlB1(close);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) == 0)&&
(DEST(DestinationC) ~= 0)
                GateCtrlA1(close);
                GateCtrlB1(open);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) ~= 0)&&
(DEST(DestinationC) == 0)
                GateCtrlA1(close);
                GateCtrlB1(close);
                GateCtrlC1(open);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) ~= 0)&&
(DEST(DestinationC) ~= 0)
```

```
                GateCtrlA1(close);
                GateCtrlB1(close);
                GateCtrlC1(close);
            end
        elseif (M(DestinationA,(3 + DestinationB)) == 1) && (M(DestinationA,(6
+ DestinationC)) == 0) && (M(3 + DestinationB,(6 + DestinationC)) == 0)
            if (DEST(DestinationA) == 0) && (DEST(DestinationB) == 0)&&
(DEST(DestinationC) == 0)
                GateCtrlA1(open);
                GateCtrlB1(open);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) == 0) && (DEST(DestinationB) == 0)&&
(DEST(DestinationC) ~= 0)
                GateCtrlA1(open);
                GateCtrlB1(open);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) == 0) && (DEST(DestinationB) ~= 0)&&
(DEST(DestinationC) == 0)
                GateCtrlA1(open);
    %Here I can decide which train goes by, A or C.
                GateCtrlB1(close);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) == 0)&&
(DEST(DestinationC) == 0)
                GateCtrlA1(close);
    %Here I can decide which train goes by, B or C.
                GateCtrlB1(open);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) == 0) && (DEST(DestinationB) ~= 0)&&
(DEST(DestinationC) ~= 0)
                GateCtrlA1(open);
                GateCtrlB1(close);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) == 0)&&
(DEST(DestinationC) ~= 0)
                GateCtrlA1(close);
                GateCtrlB1(open);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) ~= 0)&&
(DEST(DestinationC) == 0)
                GateCtrlA1(close);
                GateCtrlB1(close);
                GateCtrlC1(open);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) ~= 0)&&
(DEST(DestinationC) ~= 0)
                GateCtrlA1(close);
                GateCtrlB1(close);
                GateCtrlC1(close);
                GateCtrlP(close);
            end
        elseif (M(DestinationA,(3 + DestinationB)) == 0) && (M(DestinationA,(6
+ DestinationC)) == 1) && (M(3 + DestinationB,(6 + DestinationC)) == 1)
            if (DEST(DestinationA) == 0) && (DEST(DestinationB) == 0)&&
(DEST(DestinationC) == 0)
```

```
    %Here I can decide which trains go by, A&C or B&C.
                GateCtrlA1(close);
                GateCtrlB1(open);
                GateCtrlC1(open);
                GateCtrlP(open);
            elseif (DEST(DestinationA) == 0) && (DEST(DestinationB) == 0)&&
(DEST(DestinationC) ~= 0)
                GateCtrlA1(close);
    %Here I can decide which train goes by, B or A.
                GateCtrlB1(open);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) == 0) && (DEST(DestinationB) ~= 0)&&
(DEST(DestinationC) == 0)
                GateCtrlA1(open);
                GateCtrlB1(close);
                GateCtrlC1(open);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) == 0)&&
(DEST(DestinationC) == 0)
                GateCtrlA1(close);
                GateCtrlB1(open);
                GateCtrlC1(open);
                GateCtrlP(open);
            elseif (DEST(DestinationA) == 0) && (DEST(DestinationB) ~= 0)&&
(DEST(DestinationC) ~= 0)
                GateCtrlA1(open);
                GateCtrlB1(close);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) == 0)&&
(DEST(DestinationC) ~= 0)
                GateCtrlA1(close);
                GateCtrlB1(open);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) ~= 0)&&
(DEST(DestinationC) == 0)
                GateCtrlA1(close);
                GateCtrlB1(close);
                GateCtrlC1(open);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) ~= 0)&&
(DEST(DestinationC) ~= 0)
                GateCtrlA1(close);
                GateCtrlB1(close);
                GateCtrlC1(close);
                GateCtrlP(close);
            end
        elseif (M(DestinationA,(3 + DestinationB)) == 1) && (M(DestinationA,(6
+ DestinationC)) == 1) && (M(3 + DestinationB,(6 + DestinationC)) == 0)
            if (DEST(DestinationA) == 0) && (DEST(DestinationB) == 0)&&
(DEST(DestinationC) == 0)
    % Here I can decide which trains go by, A&B or A&C.
                GateCtrlA1(open);
                GateCtrlB1(open);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) == 0) && (DEST(DestinationB) == 0)&&
(DEST(DestinationC) ~= 0)
                GateCtrlA1(open);
                GateCtrlB1(open);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) == 0) && (DEST(DestinationB) ~= 0)&&
(DEST(DestinationC) == 0)
                GateCtrlA1(open);
                GateCtrlB1(close);
                GateCtrlC1(open);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) == 0)&&
(DEST(DestinationC) == 0)
                GateCtrlA1(close);
    %Here I can decide which train goes by, B or C.
                GateCtrlB1(open);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) == 0) && (DEST(DestinationB) ~= 0)&&
(DEST(DestinationC) ~= 0)
                GateCtrlA1(open);
                GateCtrlB1(close);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) == 0)&&
(DEST(DestinationC) ~= 0)
                GateCtrlA1(close);
                GateCtrlB1(open);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) ~= 0)&&
(DEST(DestinationC) == 0)
                GateCtrlA1(close);
                GateCtrlB1(close);
                GateCtrlC1(open);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) ~= 0)&&
(DEST(DestinationC) ~= 0)
                GateCtrlA1(close);
                GateCtrlB1(close);
                GateCtrlC1(close);
                GateCtrlP(close);
            end
        elseif (M(DestinationA,(3 + DestinationB)) == 1) && (M(DestinationA,(6
+ DestinationC)) == 0) && (M(3 + DestinationB,(6 + DestinationC)) == 1)
            if (DEST(DestinationA) == 0) && (DEST(DestinationB) == 0)&&
(DEST(DestinationC) == 0)
    % Here I can decide which trains go by, A&B or B&C.
                GateCtrlA1(open);
                GateCtrlB1(open);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) == 0) && (DEST(DestinationB) == 0)&&
(DEST(DestinationC) ~= 0)
                GateCtrlA1(open);
                GateCtrlB1(open);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) == 0) && (DEST(DestinationB) ~= 0)&&
(DEST(DestinationC) == 0)
                GateCtrlA1(open);
    %Here I can decide which train goes by, A or C.
                GateCtrlB1(close);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) == 0)&&
(DEST(DestinationC) == 0)
                GateCtrlA1(close);
                GateCtrlB1(open);
                GateCtrlC1(open);
                GateCtrlP(open);
            elseif (DEST(DestinationA) == 0) && (DEST(DestinationB) ~= 0)&&
(DEST(DestinationC) ~= 0)
                GateCtrlA1(open);
                GateCtrlB1(close);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) == 0)&&
(DEST(DestinationC) ~= 0)
                GateCtrlA1(close);
                GateCtrlB1(open);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) ~= 0)&&
(DEST(DestinationC) == 0)
                GateCtrlA1(close);
                GateCtrlB1(close);
                GateCtrlC1(open);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) ~= 0)&&
(DEST(DestinationC) ~= 0)
                GateCtrlA1(close);
                GateCtrlB1(close);
                GateCtrlC1(close);
                GateCtrlP(close);
            end
        elseif (M(DestinationA,(3 + DestinationB)) == 1) && (M(DestinationA,(6
+ DestinationC)) == 1) && (M(3 + DestinationB,(6 + DestinationC)) == 1)
            if (DEST(DestinationA) == 0) && (DEST(DestinationB) == 0)&&
(DEST(DestinationC) == 0)
    % Here I can decide which trains go by, A&B, B&C or A&C.
                GateCtrlA1(open);
                GateCtrlB1(open);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) == 0) && (DEST(DestinationB) == 0)&&
(DEST(DestinationC) ~= 0)
                GateCtrlA1(open);
                GateCtrlB1(open);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) == 0) && (DEST(DestinationB) ~= 0)&&
(DEST(DestinationC) == 0)
                GateCtrlA1(open);
                GateCtrlB1(close);
                GateCtrlC1(open);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) == 0)&&
(DEST(DestinationC) == 0)
                GateCtrlA1(close);
                GateCtrlB1(open);
                GateCtrlC1(open);
                GateCtrlP(open);
            elseif (DEST(DestinationA) == 0) && (DEST(DestinationB) ~= 0)&&
(DEST(DestinationC) ~= 0)
                GateCtrlA1(open);
                GateCtrlB1(close);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) == 0)&&
(DEST(DestinationC) ~= 0)
                GateCtrlA1(close);
                GateCtrlB1(open);
                GateCtrlC1(close);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) ~= 0)&&
(DEST(DestinationC) == 0)
                GateCtrlA1(close);
                GateCtrlB1(close);
                GateCtrlC1(open);
                GateCtrlP(open);
            elseif (DEST(DestinationA) ~= 0) && (DEST(DestinationB) ~= 0)&&
(DEST(DestinationC) ~= 0)
                GateCtrlA1(close);
                GateCtrlB1(close);
                GateCtrlC1(close);
                GateCtrlP(close);
            end
        end
    end
    if g1 == 1 || g2 == 1 || g3 == 1
        GateCtrlA1(close);
        GateCtrlB1(close);
        GateCtrlC1(close);
        GateCtrlP(close);
    end

    if r1 == 1 || r2 == 1 || r3 == 1
        GateCtrlA1(close);
        GateCtrlB1(close);
        GateCtrlC1(close);
        GateCtrlP(close);
    end
else
    GateCtrlA1(close);
    GateCtrlB1(close);
    GateCtrlC1(close);
    GateCtrlP(close);
end

DestA = DestinationA;
DestB = DestinationB;
DestC = DestinationC;
x = X;
y = Y;
z = Z;
```

Command lines for PRINCIPAL GATE CONTROL 2

```
function [u,v,w,x,y,z] =
fcn(ID1,gx1,px1,x1,gx2,px2,x2,gx3,px3,x3,gx4,px4,x4,gx5,px5,x5,px6,x6,px7,x7
,px8,x8,px9,x9,X,ID2,gy1,py1,y1,gy2,py2,y2,gy3,py3,y3,gy4,py4,y4,gy5,py5,y5,
py6,y6,py7,y7,py8,y8,py9,y9,Y,ID3,gz1,pz1,z1,gz2,pz2,z2,gz3,pz3,z3,gz4,pz4,z
4,gz5,pz5,z5,pz6,z6,pz7,z7,pz8,z8,pz9,z9,Z)

open = 1;
close = 0;
nx = x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9;
npx = px1 + px2 + px3 + px4 + px5 + px6 + px7 + px8 + px9;
ngx = gx1 + gx2 + gx3 + gx4 + gx5;
ny = y1 + y2 + y3 + y4 + y5 + y6 + y7 + y8 + y9;
npy = py1 + py2 + py3 + py4 + py5 + py6 + py7 + py8 + py9;
ngy = gy1 + gy2 + gy3 + gy4 + gy5;
nz = z1 + z2 + z3 + z4 + z5 + z6 + z7 + z8 + z9;
npz = pz1 + pz2 + pz3 + pz4 + pz5 + pz6 + pz7 + pz8 + pz9;
ngz = gz1 + gz2 + gz3 + gz4 + gz5;

if (nx + npx + ngx + X) == 0
    GateCtrl1(open);
else
    GateCtrl1(close);
end

if (ny + npy + ngy + Y) == 0
    GateCtrl2(open);
else
    GateCtrl2(close);
end

if (nz + npz + ngz + Z) == 0
    GateCtrl3(open);
else
    GateCtrl3(close);
end

if (ID1 == 1) || (ID2 == 1) || (ID3 == 1)
    GateX1(open);
    GateY1(open);
    GateZ1(open);
end
if (ID1 == 2) || (ID2 == 2) || (ID3 == 2)
    GateX2(open);
    GateY2(open);
    GateZ2(open);
end
if (ID1 == 3) || (ID2 == 3) || (ID3 == 3)
    GateX3(open);
    GateY3(open);
    GateZ3(open);
end
if (ID1 == 4) || (ID2 == 4) || (ID3 == 4)
    GateX4(open);
    GateY4(open);
    GateZ4(open);
end
if (ID1 == 5) || (ID2 == 5) || (ID3 == 5)
    GateX5(open);
    GateY5(open);
    GateZ5(open);
end

%particular
```

```
if (ID1 == 1) && ((nx + npx + ngx) == 0)
    GateCtrl1a(open);
elseif (ID1 == 3) && ((nx + npx + ngx) == 0)
    GateCtrl1a(open);
else
    GateCtrl1a(close);
end

if (ID2 == 2) && ((ny + npy + ngy) == 0)
    GateCtrl2a(open);
elseif (ID2 == 4) && ((ny + npy + ngy) == 0)
    GateCtrl2a(open);
else
    GateCtrl2a(close);
end

if (ID3 == 3) && ((z1 + z2 + z3 + z6 + z7 + z8 + z9 + gz1 + gz2 + gz3 + pz1
+ pz2 + pz3 + pz6 + pz7 + pz8 + pz9) == 0) && (Z == 1)
    GateCtrl3b(open);
elseif (ID3 == 4) && ((z1 + z2 + z4 + z6 + z7 + z8 + z9 + gz1 + gz2 + gz4 +
pz1 + pz2 + pz4 + pz6 + pz7 + pz8 + pz9) == 0) && (Z == 1)
    GateCtrl3b(open);
else
    GateCtrl3b(close);
end

if (ID3 == 5) && ((nz + npz+ ngz) == 0)
    GateCtrl3a(open);
else
    GateCtrl3a(close);
end

u = nx + npx + ngx;
v = ny + npy + ngy;
w = nz + npz + ngz;
x = ID1;
y = ID2;
z = ID3;
```