Ryan Adriansyah Mulkan

Performance evaluation tool for the expansion of a port's container network by an offshore modular platform

ME54035 – MSc Project

TU Delft

Performance evaluation tool for the expansion of a port's container network by an offshore modular platform

ME54035 – MSc Project
Theme 3 – Coordination for Real-Time Logistics

By

Ryan Adriansyah Mulkan
Student Number: 4624025

in partial fulfilment of the requirements for the degree of

Master of Science
in Mechanical Engineering track Transport Engineering and Logistics

at the Delft University of Technology

Supervisor 1 : Ir. M. B. Duinkerken
Supervisor 2 : dr. Ir. D. L. Schott
Chair : Prof. dr. R. R. Negenborn

Delft University of Technology

FACULTY MECHANICAL, MARITIME AND MATERIALS ENGINEERING
Department Maritime and Transport Technology

Mekelweg 2
2628 CD Delft
the Netherlands
Phone +31 (0)15-2782889
Fax +31 (0)15-2781397
www.mtt.tudelft.nl

Specialization: Transport Engineering and Logistics

Report number: 2019.TEL.8326

Title: **Performance evaluation tool for the expansion of a port's container network by an offshore modular platform**

Author: Ryan Adriansyah Mulkan

Title (in Dutch) **Prestatie-evaluatietool voor de uitbreiding van het containernetwerk van een haven door een offshore modulair platform**

Assignment: Graduation

Confidential: No

Supervisor: Ir. M. B. Duinkerken

Date: March 13th, 2019

Acknowledgements

This report is a documentation of the graduation project titled '*Performance evaluation tool for the expansion of a port's container network by an offshore modular platform'*. The report is one of the final requirements to complete the Masters of Science study in the track of Transport Engineering and Logistics at Delft University of Technology. This graduation project is conducted as a part of the task T9.5 under the Work Package 9 (WP9) of the Space@Sea project.

I would like to grant special gratitude to Prof. dr. Rudy Negenborn as the chair of my graduation committee who also gave me the opportunity to work on this topic. Special appreciations should be addressed to Mark Duinkerken and Dimitris Souravlias who have supervised and helped me during this project; as well as Dingena Schott as the WP-9 leader who also acts as one of my graduation committee. I also appreciate all the critics, supports, and small talks from the people of Maritime & Transport Technology Department of TU Delft who are involved in the Space@Sea project (Pieter, Giannis, Bilge, Vittorio). Lastly, I would like to mention Lembaga Pengelola Dana Pendidikan Republik Indonesia (LPDP-RI), for trusting me with a scholarship and giving me the chance to study here in The Netherlands.

Ryan Adriansyah Mulkan
Delft, March 2019

Summary

The Space@Sea is a project funded by the Horizon 2020 programme of the European Commission. The project aims to develop a sustainable offshore modular platform for a workspace at sea as an effort in coping up with the economic growth and allocating marine resources to more efficient uses. Currently, sea-going vessels have to transfer their containers at the Port of Antwerp that is located inland and connected to the sea via the Scheldt river. The presence of the platform will influence the logistics operations at this port. Therefore, this research proposes a question to be answered at the end of this report: *"How would the offshore modular platform, when implemented as a container transhipment/storage hub, affects the handling capacity of an existing port system?"*

Though the Port of Antwerp has a total of 24 terminals, only five of them are dedicated to handling sea-going container vessels. The connection between these terminals is provided by three modalities: trucks, trains, and barges. In order to promote the barge transport, some initiatives have introduced three programmes: a) central coordination point for vessels and barges, b) consolidation of small volumes, and c) the *Premium Barge Service* (PBS). The PBS is a barge shuttle service that distributes containers within the port area. The operation of the PBS could be extended to integrate the operation of the platform to the existing port service.

There are many processes involved in a container terminal. One of the most relevant processes is the *inter terminal transport* (ITT), which can be defined as transport of containers between terminals, depots and distribution centres in the area of the port with the aid of various transport modes. The important parameters, performance indicators, as well as the methodologies used in ITT studies can be found in the literature. The *discrete event simulation* method is used in this research. This method is one of the popular methodologies that is used to solve ITT problems, as it enables evaluation of various alternatives and represents the stochastic behaviour at the operational level. With this method, it is also possible to consider and evaluate several operational scenarios of the platform.

A simulation model is developed to evaluate the performance of the port and platform. The model is built based on the *Delft System Approach*. Six performance indicators are used to determine the performance of the system: a) percentages of vessels and containers that are handled at the platform, b) quay crane utilization rate of the terminal, c) vessel handling time, d) statistics of the terminal stacks, e) container dwell time, and f) non-performance rate. A vessel allocation algorithm is introduced to determine which terminal the vessels should call to. The algorithm is based on three parameters: a) capacity of the vessel, b) maximum storage capacity of the platform, and c) prediction of the number of available quay cranes in the terminals at the arrival time of the vessel. Two different route strategies are constructed for the barge shuttle service. The container handling demand is constructed based on a historical dataset of the vessels visiting the Port of Antwerp in the year 2017. The effect of the wind speed and sea waves to the productivity of the terminal are taken into account in the model. The wind and sea wave scenarios are constructed based on a historical dataset at a location in the North Sea. In order to verify and validate the model, a series of test runs and checks are conducted using the model.

Five cases are simulated in order to investigate how the port and the platform would perform with respect to different barge route and vessel allocation strategies. The preferred strategies are then used to evaluate several other configurations and scenarios, e.g., a) different number of quay cranes on the platform, b) different platform locations, c) different demand scenarios, d) different maximum storage capacity of the platform, and e) different sea states scenarios. In general, the installation of the platform will increase the total handling capacity of the port, and it will reduce the QC utilization rate and the need for storage capacity at the port terminals. Based on the simulation results, this research suggests to implement the two-barge-route strategy for the container distribution scheme between the port and the platform and recommends the platform to only handle vessels with capacity smaller than 6,000 TEUs. Finally, it is recommended for research in the future to investigate the coordination scheme with the hinterland terminals, so direct transhipment at the S@S platform can be actualized.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AGV	Automated Guided Vehicle
AIS	Automatic Identification System
ALV	Automated Lifting Vehicle
BC	Barge Crane
BTS	Barge Traffic System
CMT	Conventional Marine Terminal
CTOS	Container Terminal Operations Simulator
FCSTT	Floating Container Storage Transshipment Terminal
FIFO	First In, First Out
FMT	Floating Marine Terminal
IDE	Integrated Development Environment
IP	Integer Programming
ITT	Inter Terminal Transport
MFCT	Mega Floating Container Terminal
MTS	Multi Trailer System
PBS	Premium Barge Service
QC	Quay Crane
SC	Straddle Carrier
TEU	Twenty-foot Equivalent Unit
TOMAS	Tools for Object-Oriented Modelling and Simulations
ULCS	Ultra Large Container Ships

“For Mulkan Hamid and Surnetti Moeis,
whose always been the main reason for everything I have achieved in life.”

Chapter 1 Introduction

The Space@Sea is a project funded by the Horizon 2020 programme of the European Commission [1]. The project aims to develop a sustainable offshore modular platform for a workspace at sea as an effort in coping up with the economic growth and improving the allocation of marine resources to more efficient uses. There are four applications of the platform to be studied in the project: farming, transport & logistics hub, energy hub, and living [2]. The project was initiated in November 2017 and planned as a three-year project involving 17 project partners from all across Europe. Being one of the partners, TU Delft is responsible for the design of the transport & logistics hub and its equipment, as well as the respective operational real-time process coordination [3].

As one of the work packages in the project, the WP-9 Transport&Logistics@Sea attempts to integrate the logistics operations of the offshore modular platform to an existing port service by coordinating at the strategic, tactical, and operational level. Several studies are conducted to determine the essential decisions for the platform configuration, e.g., size of the platform, number of loading and unloading equipment, storage capacity, types of the vessels to be handled at the platform, as well as the coordination with the existing port system.

1.1 Progress of the Transport&Logistics@Sea

The WP-9 Transport&Logistics@Sea consists of eight tasks, starting from the initial collaboration of the project partners to the final demonstration of the proposed design. Currently, TU Delft is involved in five tasks, three of them listed as follow:

a) T9.3: Design of transhipment and floating storage on the platform
b) T9.4: Strategic logistics optimization on the platform
c) T9.5: Operational real-time process coordination on and around the platform

The partners from T9.3 has proposed some layout concepts of the platform. In general, the terminal at the platform will have a rectangular shape with quay cranes installed at two sides that are across of each other. The concepts are made by considering different scenarios regarding the operation of the platform terminal.

Meanwhile, the partners from T9.4 are dealing with the optimization of the platform's equipment configuration. They use operation research approaches to optimize the number of terminal equipment on the platform with respect to both the expected annual throughput and cost of the platform. It is intended that the outcome of T9.4 could support the planning and decision making on the strategic level.

A potential location of the platform is shown in Figure 1.1. One of the closest ports from this location is the Port of Antwerp. Currently, sea-going vessels have to transfer their containers at the Port of Antwerp that is located inland. The vessels need to make a turn into the Scheldt

river, queue and wait while their containers are handled at the port, then sail back to the sea to continue with their journey.

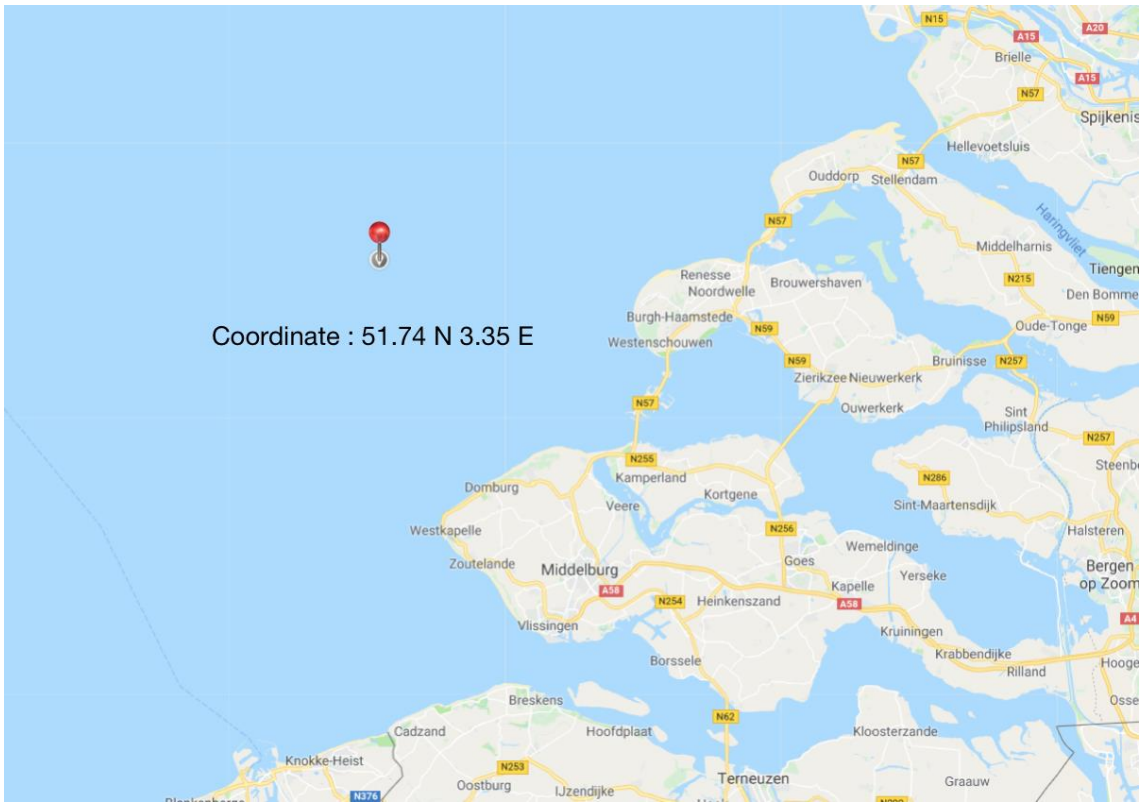


Figure 1.1 Potential location of the platform

The presence of the platform will influence the logistics operations of this port. Instead of having to turn into the river passage, the sea-going vessels also have the choice to have their containers handled at the platform, and then continue straight with their journey. A dedicated connection from the port can then pick the containers. However, due to its modular and offshore characteristics, the performance of the platform will be influenced by the sea states.

While studies on the design and configuration of the platform have been conducted, those concerning the logistics operations on and around the platform have not been initiated. There is a need to investigate how the platform would affect the performance of the existing port, and how the platform should coordinate with the port when different configurations and scenarios are implemented; for instances:

1) If the platform only handles small & medium vessels (100 – 6,000 TEUs),
2) If the platform only handles large vessels (≥ 6,000 TEUs), and
3) If the platform only handles Ultra Large Container Vessels.

1.2 Problem statement

Based on the conditions described beforehand, the main question to be answered at the end of this research is stated below:

"How would the modular offshore platform, when implemented as a container transhipment/ storage hub, affects the handling capacity of an existing port?"

Several sub-questions are created to establish a firm guideline in answering the main question:

a) What are the components of the port? How are they configured, and how do they currently perform? Who are the stakeholders, and what kind of services do they provide for handling incoming containers?
b) What are the processes involved during the handling of containers within a port? What are the performance indicators? What are the common methods that are used to evaluate the performance of a port? In terms of how they handle containers, what are the differences between a land-based terminal and an offshore terminal?
c) How should the port and the platform be modelled, and how should the platform coordinate with the port in order to have an integrated container handling service? What are the configurations and scenarios to be evaluated using the model? Does the model behave the way the user intended it to be?
d) Does the model represent the behaviour of the real port? If so, how do the different configurations and scenarios affect the port's and platform's performance?

1.3 Contribution of this research

This research will develop a tool to evaluate the performance of a port which container handling service is extended by the presence of an offshore modular platform. The tool can be used by the S@S partners to have insights on how the S@S platform affects the handling capacity of the Port of Antwerp when the configurations and scenarios proposed in T9.3 and T9.4 are actualized. Though this research uses the Port of Antwerp as a case study, the tool is generic enough to be used by other port authorities to compare the performance of a port when, for example, a new intra-port container distribution scheme is implemented, or when a new transhipment/storage hub is introduced within the port network.

Despite the fact that there has been a lot of research regarding the operation of an offshore terminal, they are not focused on how to coordinate the offshore terminal operation to an existing container handling service. Moreover, most of the research that looks into the operational aspects of an offshore terminal only investigated the terminal's and vessel's motion responses due to the sea wave motions. There is still a limited number of research on how these motions affect the overall handling capacity of the extended port service. Therefore, this research will try to contribute to the scientific domain by filling these literature gaps.

1.4 Approach

This research starts with the analysis of the Port of Antwerp and its components. A literature review is conducted afterwards as an effort to find theoretical backgrounds and state-of-the-art of the topics that are relevant to this research. Then, a model is developed along with several configurations and scenarios of the port-platform logistics operations. Finally, some experiments are conducted using the model, and the results of these experiments are analyzed to determine the effect of the platform installation to the handling capacity of the port.

1.5 Structure of the report

This report will be structured in a way that the underpinning questions are consequently answered. Chapter 2 consists of the system description and analysis. Chapter 3 discusses the latest literature with topics related to the issues addressed in this research. Chapter 4 describes the development of a model and several configurations and scenarios. Chapter 5 provides an experimental plan and discussions on the experiment results. Finally, Chapter 6 concludes the report and suggests some recommendations for future research.

Chapter 2 System Analysis

This chapter provides the answers to these sub-questions: *"What are the components of the port? How are they configured, and how do they currently perform? Who are the stakeholders, and what kind of services do they provide for handling incoming containers?"*. Section 2.1 gives general information of the Port of Antwerp, while Section 2.2 describes the current configuration of the port's infrastructure. Following the previous two, Section 2.3 elaborates the Instream initiatives and their inland shipping programmes. Finally, Section 2.4 summarizes the chapter.

2.1 Port of Antwerp overview

The Port of Antwerp is an international seaport which has acted as one of the major links of the world trade since the beginning of the 19th century. The port covers a total area of 12,068 hectares at the banks of the Scheldt River. This area is used by about 900 private companies to perform their logistics activities. Meanwhile, the port infrastructure is managed by the Antwerp Port Authority. The port handles five types of goods: containers, liquid bulk, dry bulk, breakbulk, and ro-ro.

The layout of the port is shown in Figure 2.1. An enlarged version of the port map is included in Appendix B. The area is divided by the Scheldt river, and the port terminals are located on both the left and right river banks. There are 24 terminals at the port in total. The operators of these terminals are listed in Table 2.1. As shown in Figure 2.1, these terminals are grouped into three clusters: the left river bank side (cluster 1), the outer part of the right river bank side (cluster 2), and the inner part of the right river bank side (cluster 3).

Data of container handling in Port of Antwerp is available from the port's yearly statistic book [4]. With an overall performance rate of 40 crane movements per hour per crane, the Port of Antwerp by far has the most productive deep sea container terminals throughout Europe [5]. The port was visited by a total of 4,289 container vessels and 59,268 hinterland barges in 2017 [4]. Within the same year, containers took about 55% from a total of 223 million tonnes of maritime freight volume handled in the port. As shown in Figure 2.2, this percentage has approximately remained the same since 2010; but the actual number of containers has increased exponentially throughout 30 years. In addition, the port is also accessible for the *Ultra Large Container Ships* (ULCS), which is the current biggest container vessels in the world [6], [7].

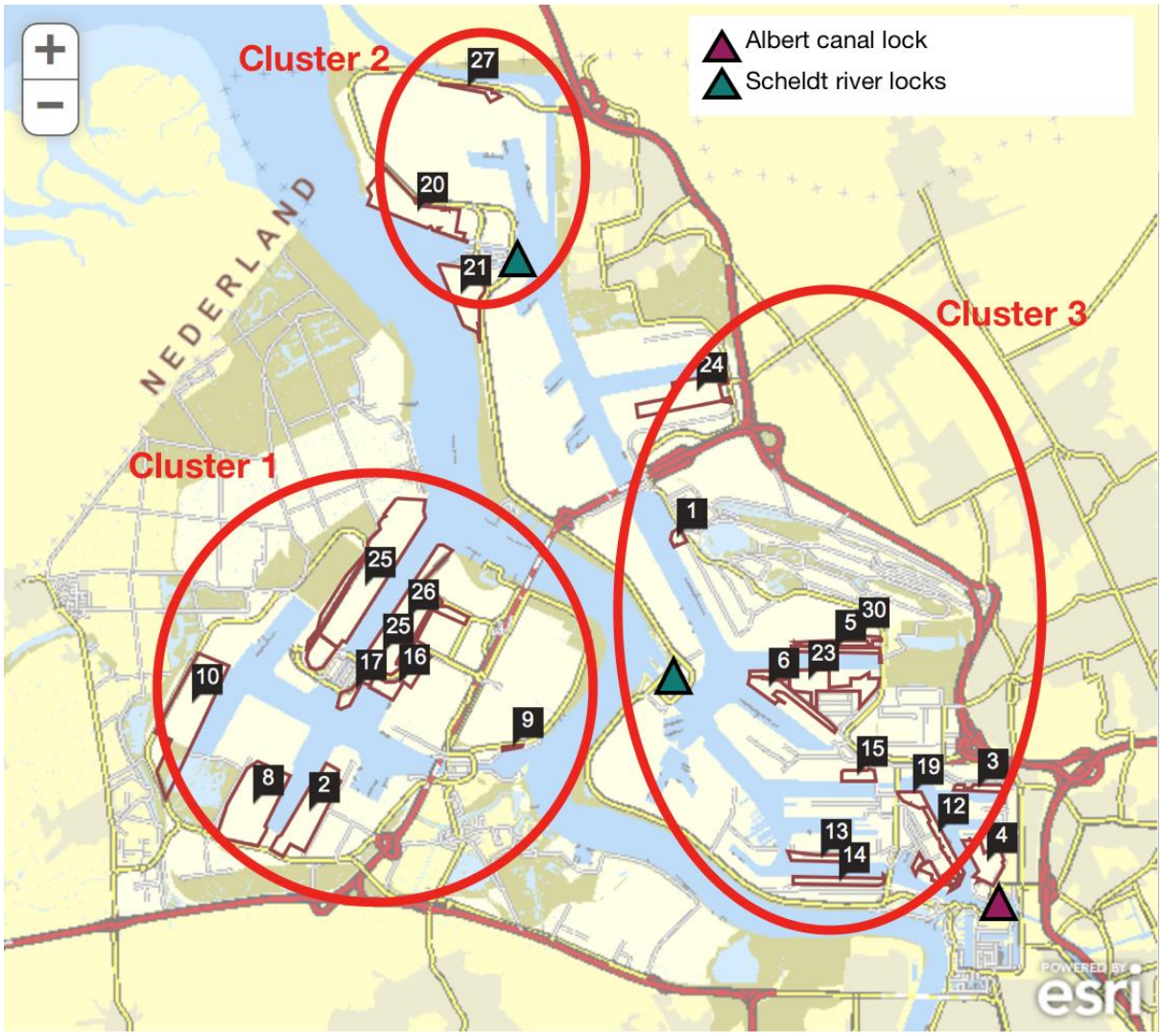


Figure 2.1 Terminal clusters at the port [5], [8]

Table 2.1 Names of the terminals [9]

#	Terminal Name	#	Terminal Name	#	Terminal Name
1	Euroports Containers 524	10	Antwerp Euroterminal NV	20	PSA Antwerp Noordze Terminal
2	Euroports Antwerp Leftbank K1207	12	Wijngaard Natie Terminals	21	PSA Antwerp Europa Terminal
3	Euroports Terminals Antwerp K168	13	Mexiconatie NV	23	PSA Antwerp Churchill Terminal
4	Terminal Zuid	14	Katoen Natie Terminals NV	24	Independent Maritime Terminal
5	Zuidnatie Breakbulk	15	BNFW	25	MSC PSA European Terminal
6	Associated Terminal Operators	16	DP World Antwerp GCS	26	DP World Antwerp Gateway
8	Katoennatie Terminals	17	Shipit	27	Combinant NV
9	Katoennatie Terminals K1510	19	AST	30	Hupac Intermodal BVBA

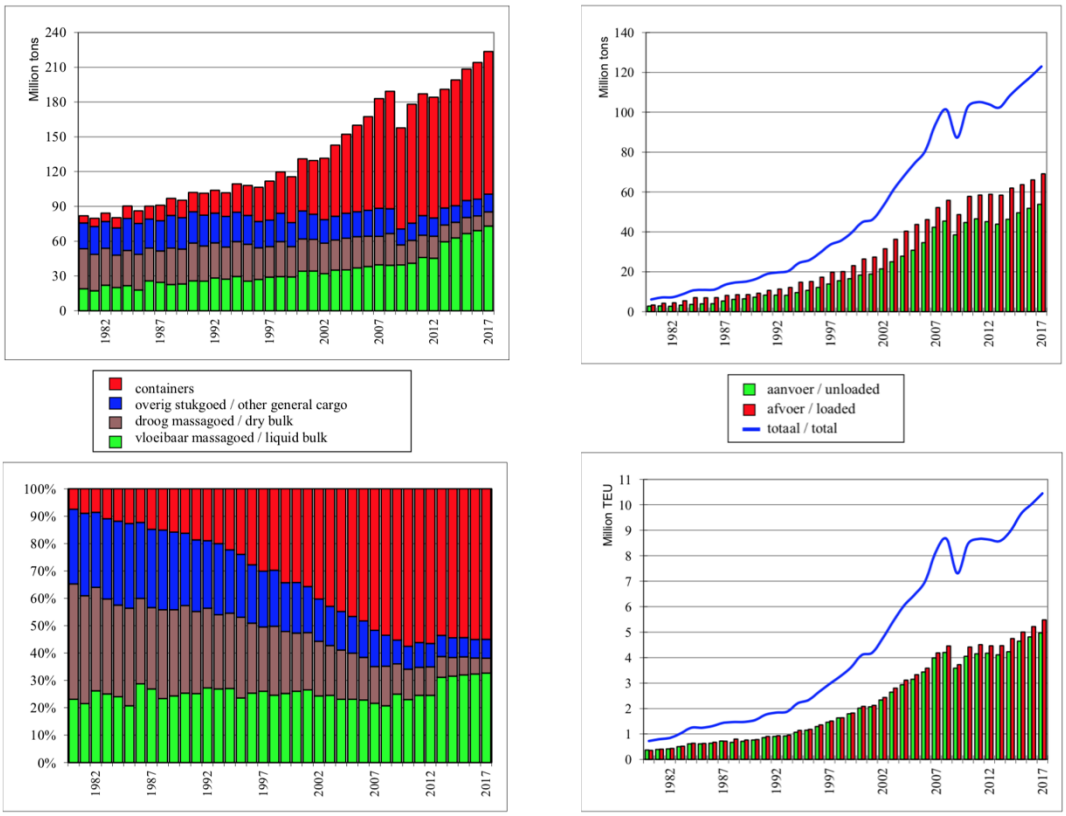


Figure 2.2 Increase of container handling demand at the Port of Antwerp [4]

2.2 Port infrastructure

The container facilities of the port (i.e. customs, empty depots, repair centres, etc.) are spread along with the 24 terminals. However, sea-going container vessels can only be handled at the five maritime container terminals. All the containers coming and leaving to/from Port of Antwerp must have been handled in one of these terminals. Moreover, it is foreseen that in the future these five terminals will only handle vessels with call size more than 30 containers [10]. These terminals are: a) MSC PSA European Terminals, b) DP World Antwerp Gateway Terminal, c) PSA Noordzee Terminal, d) PSA Europa Terminal, and e) Independent Maritime Terminal. The specifications of these five terminals are listed in Table 2.2.

Table 2.2 Specification of the container terminals [11]–[14]

Terminal name (ID)	Capacity (x10⁶ TEUs)	Quay length (m)	# of quay cranes
PSA Noordzee Terminal (T20)	2.2	1,125	12
PSA Europa Terminal (T21)	1.8	1,180	8
MSC PSA European Terminals (T25)	9	3,700	41
DP World Antwerp Gateway Terminal (T26)	2.8	1,660	11
Independent Maritime Terminal (T24)	1.89	1,050	3
Total	17.70		

2.2.1 Hinterland and intra-port connections

With a central location and well-developed hinterland connections, the port can deliver containers to their next destinations by road transport, rail transport, or inland shipping. Most of the port terminals have multi-modal hinterland connections. These three modes are also used for the intra-port container distribution. Therefore, the container flow in Port of Antwerp can be illustrated as shown in Figure 2.3.

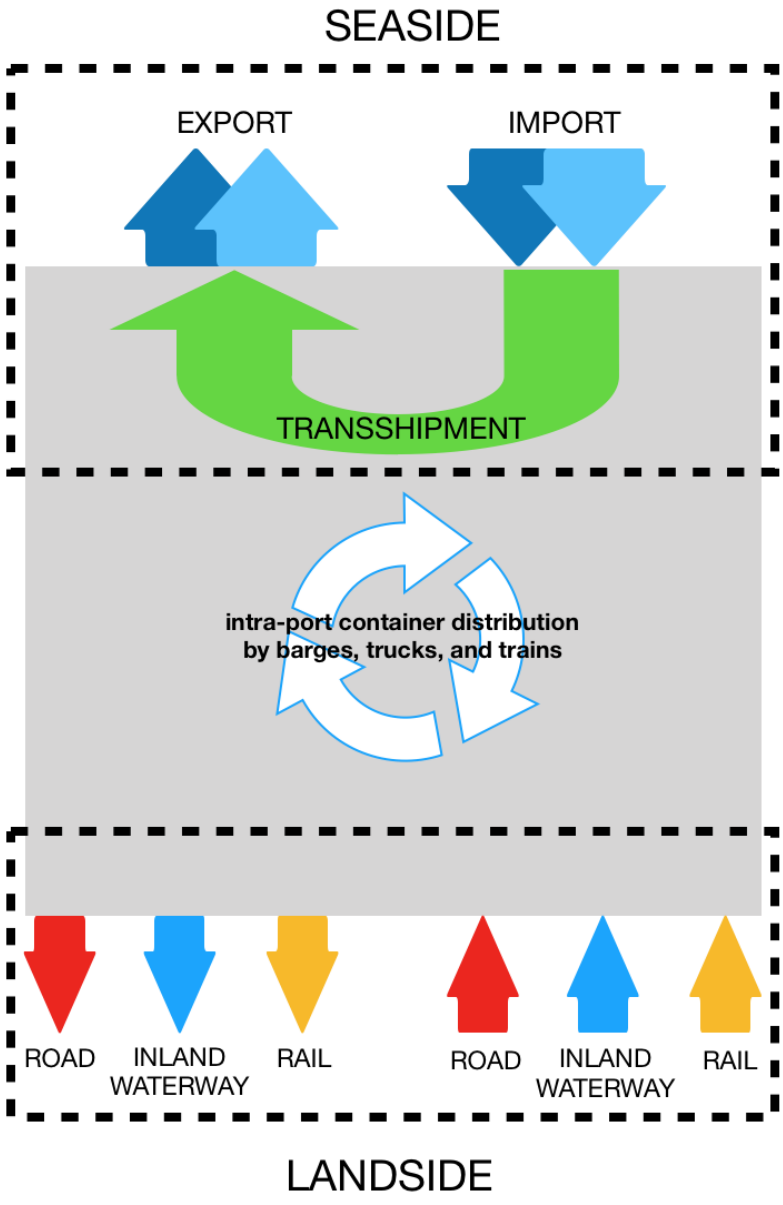


Figure 2.3 Container flow at the Port of Antwerp

2.2.2 Modal split

The current and prospected modal split of Port of Antwerp is shown in Figure 2.4. A comparison to the modal split of Port of Rotterdam is given in Figure B.2. Currently, the Port of Antwerp is trying to reduce the share of road transport due to sustainability reasons [15]. In general, the percentage of goods transported by road transport and inland shipping should be reallocated to rail transport. However, specifically for containers, this reallocation of the road transport share should be prorated to both the rail transport and inland shipping. Stakeholders from both transport modes have been working collaboratively to improve and optimize the operations of these transport modes.

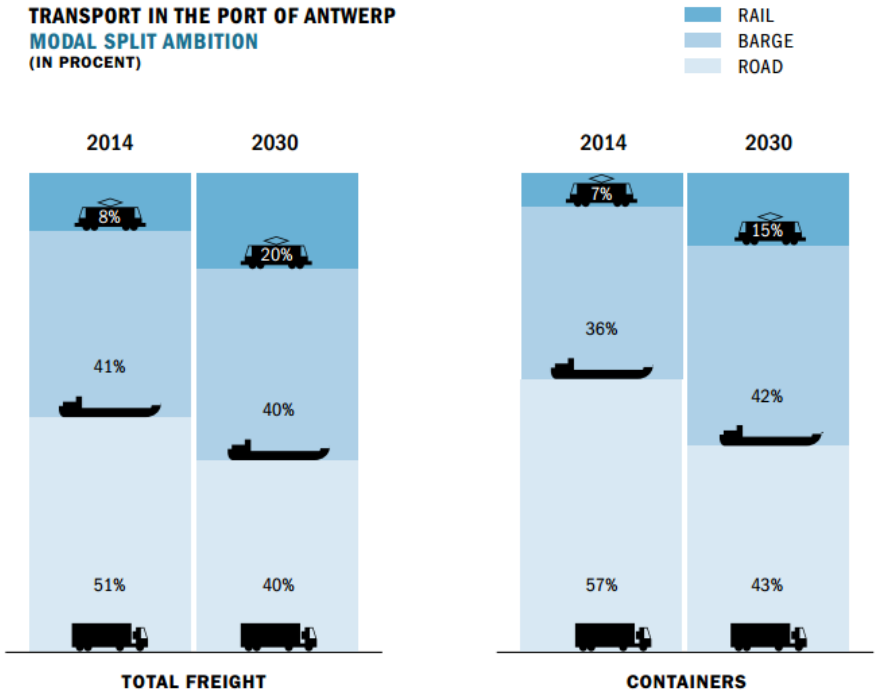


Figure 2.4 Modal split in Port of Antwerp [15]

2.2.3 Quay cranes & barge cranes

The equipment that are used to load/unload containers from vessels are called quay cranes (QCs). The size and specification of a QC may vary, depending on the needs of the terminals. Nowadays, QCs are designed so that they can handle the biggest container vessels. A picture of QCs in a terminal is shown in Figure 2.5.

.

Figure 2.5 QCs of the Port of Antwerp (source: Google Image)

Apart from the QCs, the Port of Antwerp also has smaller cranes that are used specifically to handle the barges. These *barge cranes* (BCs) are not only smaller in size, but also have more variations due to their specific uses. They can be static or mobile. Some examples of BCs are illustrated in Figure 2.6.

(a) Reach-stacker (b) Mobile BC

Figure 2.6 Different kinds of BCs (source: Google Image)

2.3 Instream: inland shipping initiatives

Instream is a collaborative inland shipping programme between the Port of Antwerp and other close partners initiated to achieve the port's modal split ambition [8]. Three main concerns are being addressed in the programme: a) *nautical coordination*, b) *efficient container handling*, and c) *effective container distribution within the port*.

As an effort to improve the nautical coordination, the *Automatic Identification System* (AIS) is implemented to locate vessels that are sailing around the port. Meanwhile, the *Barge Traffic System* (BTS) allows barge and terminal operators to communicate effectively while allocating time slots of the terminals. A central coordination point is used to coordinate the call schedules of all the port terminals. In order to have less call at the five maritime terminals, the Instream initiatives have introduced a consolidation hub in the port area. Vessels with call sizes smaller than 30 moves can consolidate their volumes at this hub [10]. Moreover, the Instream initiatives also introduce the *Premium Barge Service* (PBS), which is an hour-based barge shuttle service that allows inter-terminal container transportation inside the port area. Since 2013, the PBS is provided by Antwerp Port Shuttle N.V. [16], which is a joint company of three barge operators in the port. Table 2.3 shows the schedule of the PBS. The route of the PBS is shown in Figure 2.7. Apart from the five maritime terminals, the PBS also visit quay K364, which is one of the consolidation hubs in the port area.

Table 2.3 Schedule of the intra-port barge shuttle service [16]

Quay ID	Time at quay
K913 (Noordzee)	06.00
K869 (Europa)	08.30
K1742 (MPET)	10.30
K1700 (DP World Antwerp Gateway)	13.00
K364 (ATO)	18.30
K730 (IMT)	22.00

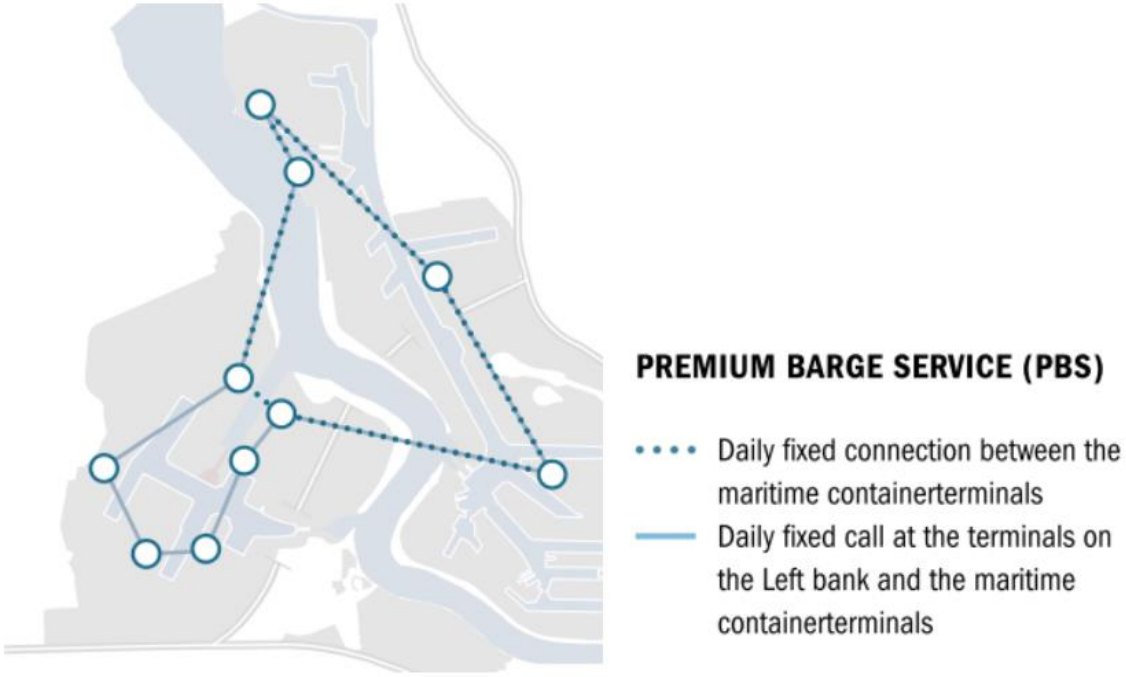


Figure 2.7 Route of the intra-port barge shuttle service [8]

2.4 Summary of the chapter

This chapter has observed the Port of Antwerp as the research's object of interest. Section 2.1 answers the first sub-question by giving an overview of the port. Section 2.2 answers the second sub-question by getting more into detail to the port terminals' specification, the hinterland connections, and the prospected modal split of the port. Lastly, Section 2.3 answers the third sub-question by bringing up the Instream initiatives. These initiatives have introduced the PBS as the barge shuttle service that distributes container within the port area.

Chapter 3 Literature Review

This chapter will look into the literature to answer these sub-questions: *"What are the processes involved during the handling of containers within a port? What are the performance indicators? What are the common methods that are used to evaluate the performance of a port? In terms of how they handle containers, what are the differences between a land-based terminal and an offshore terminal?"*. Section 3.1 gives an overview of the processes in a container terminal. Section 3.2 discusses the inter terminal transport and the methodologies that are used to analyze the performance of a port. Section 3.3 examines container handling operations on a floating container terminal. Finally, Section 3.4 summarizes the answers to the sub-questions.

3.1 Processes of container handling in a terminal

In 2003, Vis and de Koster [17] made an overview of the handling processes in container terminals. The processes are illustrated in Figure 3.1.

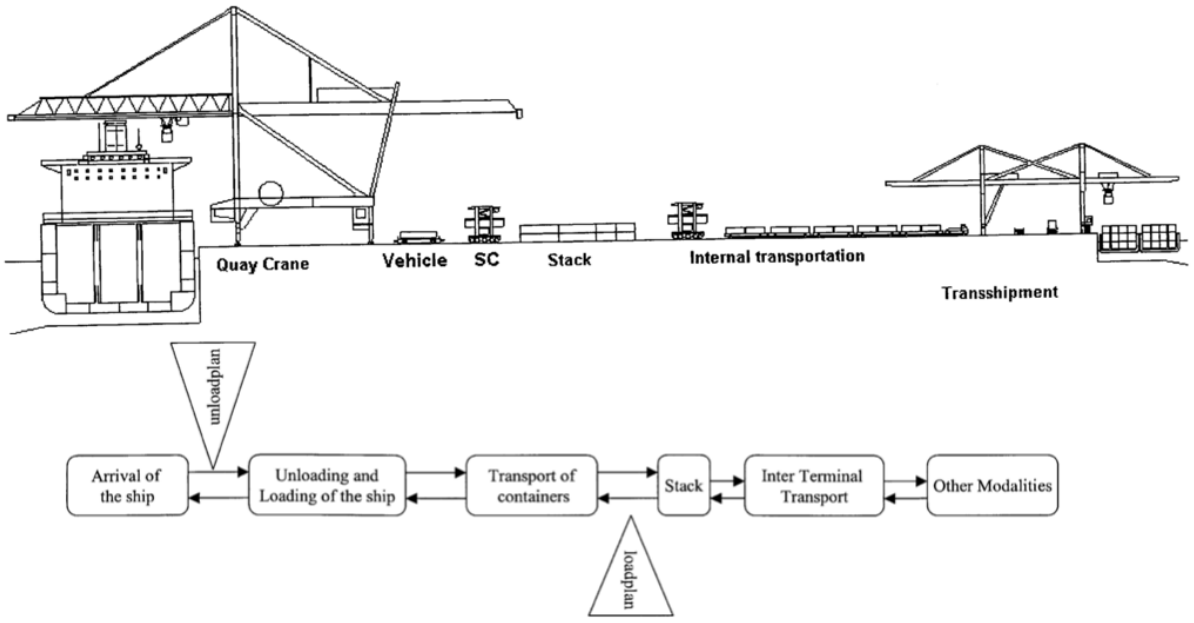


Figure 3.1 Processes of container handling in a terminal [17]

In general, there are five main processes in a container terminal, which are described as follow:

a) *Vessel arrival* — When a vessel arrives at a terminal, a part of the terminal's berth is allocated for the vessel, and some QCs are assigned to the vessel. The most common research topic regarding this process is the *berth allocation problem*.
b) *Unloading/loading of the vessels* — In a conventional container terminal, the process of unloading a vessel depends heavily on the preference of the QC driver. In this case, the unloading time has a significant variance and thus hard to study. In contrast, there is a lot of research on the loading process of a vessel. The most common research topic regarding this process is the *quay crane scheduling problem*.
c) *Transport of containers from quayside to stack and vice versa* — There are various kinds of vehicle that can be used to perform this process: forklifts, *multi-trailer system* (MTS), *straddle carriers* (SCs), *automated guided vehicles* (AGVs), and *automated lifting vehicles* (ALVs). Each of them has its performance characteristics. One example of the tactical level decision that has to be made regarding this process is the amount of vehicle needed in the terminal. Meanwhile, some strategic decisions regarding this process are the routing of the vehicle and the sorting of the containers.
d) *Stacking* — The *stack* is the area where containers are put in rows and on top of each other while they wait to be further handled. This process can be done with vast options of equipment and configurations, which are determined at the strategic level. Also, a container handling schedule should be made at the operational level in order to allocate the terminal's storage yard efficiently.
e) *Inter terminal transport (ITT) & other modalities* — Containers can be delivered either to the hinterland or the other port terminals by three transport modes: road transport, rail transport, and inland shipping. Nowadays, container terminals have multimodal hinterland connections. Just like the previous processes, most of the research regarding these two processes discuss the types and configurations of the vehicle that are used in these two processes.

This research deals with how to coordinate the logistics operations between a port and an offshore modular platform. Transport between terminals is an essential matter when it comes to how to coordinate the platform's logistics operation to the port's container handling service. Therefore, the ITT will be discussed further in the next section.

3.2 Inter Terminal Transport

The term ITT was introduced when Ottjes et al. [18] proposed a model to simulate container handling processes in port terminals. The study was done in the year 1996 as a part of the Incomaas project. The study defined the term ITT as *'the transport of containers between terminals, depots and distribution centres in the area of the port with the aid of various transport modes: railways, roads, inland shipping, and sea'*. Since then, the term started to appear in the literature; particularly in those concerning the expansion of the Maasvlakte area in Port of Rotterdam.

The latest literature of ITT can be found within the project 'Innovative concepts for Inter Terminal Transport on Maasvlakte 1 and 2 at the Port of Rotterdam'. Three institutions initiated the project: Port Research Center Delft, Erasmus SmartPort Research Cooperation, and Port of Rotterdam Authority [19]. This project aims to organize the increasing internal container transport in the Maasvlakte area by implementing innovative solutions for the ITT system. The project was conducted in seven steps which all are focused on a problem either at the strategic, tactical, or operational level. These steps are listed below:

a) Definition of the parameters of the ITT system [20]
b) Demand scenario analysis [21]
c) Generation of future demand scenarios
d) Determination of the ITT configuration [22]
e) Asset light configuration [23]
f) Cost saving analysis
g) Operational evaluation [24]

Although this framework was designed specifically for the Maasvlakte area of the Port of Rotterdam, these steps are generic enough to be used as a guideline to evaluate the expansion of another port.

3.2.1 Parameter definitions and performance indicators of ITT

Evaluating the performance of a container terminal along with all of its processes is a complicated work, as processes in container terminals have a lot of parameters and variables. Negenborn and Duinkerken [20] proposed a set of parameters that are commonly used in ITT evaluation and classified these parameters into six categories: a) *terminals*, b) *intersections*, c) *roads*, d) *vehicles*, e) *equipment*, and f) *container demand*. The parameters and their categorizations are presented in Table 3.1.

Table 3.1 Parameters of ITT [20]

Category name	Parameter(s)
Terminals	terminal number, terminal name, player name, backdoor connection(s), equipment type, # of cranes, # of QCs, yearly throughput capacity
Intersections	intersection name, type (3-way/4-way/traffic light/water), crossing time, green/red light time, capacity (# of vehicle/time)
Roads	road name, type (land/water), length, start node, start orientation, end node, end orientation
Vehicles	vehicle type, vehicle capacity, manned/unmanned, average speed, length, purchase costs, fixed costs, wage costs, fuel costs, penalty costs, mooring time, crossing factor (factor for crossing time calculation)
Equipment	equipment type, load/unload time, handling capacity (container/time)
Container demand	TEU (1 or 2), origin name, destination name, start time, delivery time, arrival/departure batch IDs (ID for grouping of the containers based on the time and locations)

Having a top priority to deliver the containers in time, the study of the ITT system in Maasvlakte 1&2 proposed the *non-performance rate* as the main performance indicator of an ITT system. The non-performance rate denotes the situation when a container arrives too late at its destination [22]–[24]. Though this indicator was used as the main performance indicator in the project, they also proposed some other indicators to provide more insights on the system performance, such as: a) the occupation rate of both the handling equipment and vehicles, b) the waiting times at the terminals, and c) the average delay of the late containers. Meanwhile, another research [25] used the waiting time and turnaround time of the inland barges as its main performance indicator.

To compare the performance of the port when the platform is installed and not installed, this research will use several performance indicators: 1) the percentage of vessels and containers that are handled at the terminals, 2) QC utilization rate of the terminals, 3) the handling time of the vessels, 4) the number of containers in the terminals' stack, 5) the dwell time of the containers in the terminals' stack, and 6) the non-performance rate of the terminals.

3.2.2 Methods to study ITT

Most container terminal problems are evaluated with either *analytical* or *simulation methods*. In their review, Vis and de Koster [17] highlighted some differences between these two methods. The analytical methods interpret the problem as a mathematical model. Then, the

model is assigned with input from a prior data collection. Finally, the solutions are derived from solving the mathematical problem. *Integer programming (IP)*, *queue models*, *network models*, and *assignment problems* are some examples of the analytical methods. Generally, these methods simplify a real-scaled problem in order to reduce the computation time in solving the problem. While these methods help in making decisions at both strategic and tactical levels, it could not represent problems of the operational level due to its simplified features. In contrast, in simulation methods, every process and factors are addressed as detailed as they can be. As a consequence, it will take a long time to construct and validate a simulation model. Yet, simulation methods have several advantages when compared to analytical methods in terms of being a decision support tool at the operational level:

- Simulations enable user to evaluate various operation alternatives [24], [26],
- They are capable of depicting the stochastic nature of container terminals [17], [27],
- The support for the animation extension to visually illustrate and verify the behaviour of the modelled system [28].

Huang et al. [29] compared these two methods in an effort to planning a container terminal. Their study aimed to determine the optimal number of berths and QCs for a container terminal by using both queuing theory and simulation model. In the experiment with queuing theory, two scenarios were analyzed. In the first scenario, it is assumed that the ships and berths are not classified according to their size and length. In the second scenario, these two aspects are taken into account. Meanwhile, a simulation model was developed using the FORTRAN language. The study stated that the outcomes of the queuing theory tend to either underestimate or overestimate the performance of the system, while the outcomes of the simulation lie in between the outcomes of the two scenarios. Though, the differences between them are minimal. Nevertheless, both methods are still used in the study of ITT, and research on both sides are still growing.

The latest literature of both methods was found in the deliverables of the Maasvlakte 1&2 project. Tierney et al. [22] provided the first, most complete IP model of ITT system to analyze container flows in expanding seaports. They used a network model with 8 terminal nodes to analyze four different transport demand scenarios. Several ITT system configurations were evaluated and compared. These configurations are: AGV, ALV, MTS, and combinations of each of the equipment with barges. The model incorporates an integrated traffic model, so vehicle congestions are taken into the evaluation. The developed model can be used to determine the optimal ITT configuration. They suggested using the model and the proposed configurations to define sets of input for a simulation model. This model was also used by Nieuwkoop et al. [23] to develop another analytical model as an effort to evaluate the operation costs of the proposed ITT system. They modelled the network into 5 clusters of terminals, which is more straightforward than the model used in [22]. Three yearly demand scenarios were compared to determine the most efficient ITT configuration out of these four proposed configurations: AGV, ALV, MTS, and a combination of barges and trucks.

As a response to the future research recommendation in [22], Schroer et al. [24] evaluated the performance of the ITT configurations with a *discrete-event simulation model*. The simulation model was made based on the same framework that had been developed before by Duinkerken et al. in 2007 [27]. Compared to the model developed in [26], the models developed in [24] and [27] are more focused on the aspects of inter terminal transport rather than quay transport. Hence, these models are more capable of evaluating the performance of each proposed ITT configuration. Although having the same framework with its predecessor, the model developed by Schroer et al. has some major improvements. The modelled ITT network consists of 18 container terminals and depots, which represent the real configuration of the Maasvlakte 1&2 area in Port of Rotterdam. An extra ITT vehicle configuration was introduced (trucks and barges), and more realistic demand scenarios were used as input for the simulation. Moreover, the newer model has successfully addressed the effect of traffic density on the route choice decisions of the ITT vehicles by incorporating a dedicated traffic model in the simulation. Schroer et al. then recommend future research to use this model to investigate the effect of different routing strategies of the inland barges to the overall performance of the ITT system.

3.3 Container handling on an offshore platform

The concept of a container transhipment terminal on an offshore platform is not something new. Researchers in Japan proposed the *Mega Float Container Terminal Facility* (MFCT) concepts in 2004 to extend the handling capacity of existing container ports. One of the concepts is to implement the MFCT as a logistics base at the outer sea area [30]. In 2005, a master student from TU Delft proposed the floating transhipment container terminal concept in his graduation thesis [31]. The thesis consists of operational and financial feasibility studies of different floating terminal concepts, configurations, and different hydrodynamical scenarios. The scenarios are constructed from the wind and sea wave conditions at several points in the North Sea. In 2012, Kim and Morrison [32] made a classification of offshore terminal concepts and studied the economic feasibility of these concepts. Later in 2013, a technical and cost evaluation on different concepts of *floating container storage and transhipment terminal* (FCSTT) is presented in [33]. The study reviewed several existing offshore floating terminals, floating storage configurations, and different kinds of floating terminal handling equipment. The study also proposed some design concepts of FCSTT by combining the different configurations and types of equipment. The study opted for the barge-structured platform configurations and suggested to use either rail-mounted slewing cranes or pedestal slewing cranes to handle the incoming container vessels. The proposed concept is based on the application of Gottwald's open-sea floating crane-barge system in Indonesia.

From the operational perspective, both [30] and [31] concluded that the container handling operation of the floating terminal is influenced by the relative motions between the floating terminal and the vessel. Based on [31], container handling operation on a floating terminal could be actualized with significant sea wave height up to 3 meters. Another study [34] compared the performance of *floating marine terminal* (FMT), or *floaterm*, with the performance of *conventional marine terminal* (CMT) under normal and disruptive conditions. This study uses similar performance indicators as what has been discussed in Section 3.2.1. However, the focus of the study is more to compare the differences between the FMT and CMT; not to show how the FMT relieves the demand for container handling at the CMT.

3.3.1 Effect of sea states to QC handling efficiency

To determine the operational limit of a QC in a land-based terminal, one can refer to the standards given by the Nederlands Normalisatie-Instituut [35]. Though the main purpose of these standards is to give a guideline in the design phase of a QC, the standard can also give a different perspective on the limitation of a QC's operation. According to the standards, wind conditions are the most influential factors to the load scenarios of a QC. The definition of the wind conditions is categorized based on wind speed, direction, and pressure. For example, a storm is defined as a condition when the wind speed is higher than 20 m/s at 10 meters above the ground. The categorization is shown in Table 21 of Appendix F.

As stated in [36], depending on what factors are considered, the productivity of a QC can be determined based on several viewpoints:

a) *Technical crane productivity* (TCP) — this notes the theoretical performance of a QC when the loading/unloading process of a container is not disrupted at all.
b) *Operational crane productivity* (OCP) — this measure takes into account operational losses such as delays due to sways and stacker handling.
c) *Net crane productivity* (NCP) — sometimes the operation of a QC is affected by the other component of the port system, e.g. the late arrival of terminal equipment. Such factors are affiliated in this rate.
d) *Gross crane productivity* (GCP) — factors such as bad weather, personnel breaks, shift changes are taken into account in this rate.

The relation between these productivity rates is shown in Figure 3.2. The *vessel handling time* is a sum of QC's GCP and some other delays that are not accountable to the terminal operator.

Meanwhile, the *berth time* is the vessel handling time plus the waiting time of the vessels at the quay.

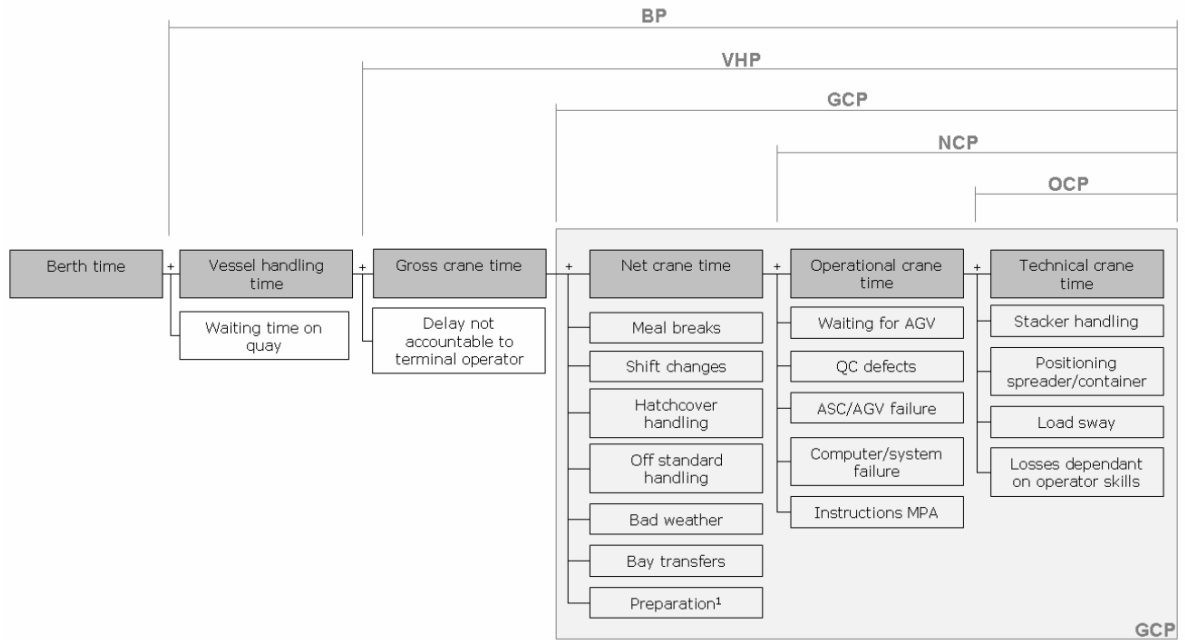


Figure 3.2 Measures of crane productivity [36]

In his thesis, Van der Loeff [36] measured the factors that contribute to a QC productivity loss. The thesis showed that bad weather took only about 0.01% of the overall loss in a month production time. However, it is important to note that the period of the data sample used in this study was in the summer period. This percentage might increase in a different period of the year when the weather is not very favourable. In 2016, Chhetri et al. [37] simulated the effect of extreme sea states on container terminal operation in Australia. This study only considered one terminal instead of a whole port, as the focus was to find out how the terminal's handling performance is affected by extreme weather scenarios. A simulation model was built and named as the *Container Terminal Operations Simulator* (CTOS). The model is an agent-based simulation model developed under a Java-based software package. The performance indicators of the model are: a) QC productivity, b) SC productivity, c) truck queue length, and d) yard utilization. The study concluded that heavy rain and strong wind are the factors that affect crane productivity the most. In 24-hour timeframe, a 6-hour heavy rain or strong wind will reduce the daily crane throughput by 13%.

On the other hand, the dynamics of a floating QC is not only affected by the wind/weather conditions, but also by the sea wave motions. There are six modes of motions of the floating terminal and the vessel: a) *surge*, b) *sway*, c) *heave*, d) *roll*, e) *pitch*, f) *yaw*. The response of a floating QC to the sea wave motions is evaluated in [38] by taking into account the heave, surge, and sway motions at the tip of the crane's boom. However, since the floating QC is designed for an extension of a terminal's berth, the floating QC is assumed to be installed on an individual pontoon-shaped structure rather than a dedicated floating terminal. In this study, the heave and surge motions are large, but the time interval between the peaks are also wide. Thus, these two motions can be easily compensated by the operator. Meanwhile, the sway motion is considered to be more difficult to be compensated.

In [30], the QC handling efficiency of the MFCT is determined based on a) the reduced acceleration of the grab trolley, and b) the displacement of the container that is being (un)loaded. Both are caused by the oscillation of the MFCT due to the sea wave motions. The study showed that the effect of the oscillation of the MFCT to the QC operation is negligible. On the other hand, Ali et al. [31] concluded that due to the large dimensions of the floating container transhipment terminal, the sea motions would not significantly influence the handling efficiency of the QCs. Moreover, another study has proposed a dedicated control

system for mobile/offshore cargo handling designed to reduce the QC's motion responses due to the sea wave motions [39].

3.3.2 Effect of sea states to vessel motions and sailing speed

The sea states, such as the wind speed and the significant sea wave height, would affect the motions of the vessels that are being handled at a terminal, as well as the sailing speed of the vessels. As indicated in [40], when the sea states are uncompromising, the maximum amount of containers that can be carried by the vessels decreases, resulting in a demand fluctuation that goes along with the seasonal changes. Meanwhile, Sukeyasu et al. [30] refer to PIANC standard to determine the allowable vessel motions in order to be handled by the QCs on the MFCT. On the other hand, Ali et al. [31] considered the vessels' heave and roll motions to be the most influential modes of motions during the handling process at a floating terminal.

Another study [41] evaluated the (un)loading process of a super container vessel from a floating quay during rough weather conditions. In this study, the floating quay is implemented as a seaside berth extension, so vessels are handled in between the floating quay and the landside quay. Thus, a numerical three-body diagram analysis is used to calculate the vessel's motion responses due to the sea wave motions. In order to validate the findings, the numerical analysis result was compared to an experiment result [42]. The study concluded that even though the floating characteristics increases the relative motion responses between the vessels and the quay, the responses are still within acceptable limits.

Meanwhile, the sailing of the vessels is also affected by the sea states. The effects have been studied since 1998 in [43]. The study constructed a traffic simulation model of the Port of Antwerp waterway. The model was developed by using SIMAN programming language and Arena software. Tides and weather conditions were included as components in the model. Data of the sea wave heights and relevant weather patterns such as strong wind, heavy rain, and fog at multiple points along the waterway were treated as input for the sailing process of the vessels. The vessels can be *tide-dependent* or *-independent*, depending on their size. Tide-dependent is when the sea wave height is higher than the vessel's draught. In this scenario, the vessels have their *tidal window*, which represents the limited period when a vessel can sail during high tides. As the main performance indicator of the model, the service time of the vessels at the port is also size-dependent and classified into five categories. The developed model has been used as a decision tool for expansion and renovation projects at the port area.

3.4 Summary of the chapter

This chapter has investigated the recent literature to find a firm theoretical background on the topics related to this research. Answers to the second group of sub-questions are elaborated. At first, the processes of container handling in a terminal are described in Section 3.1. Then, the important parameters and key performance indicators for the intra-port container distribution, or ITT, are elaborated in Section 3.2. This section also gives a comparison between the two groups of methods that are used to evaluate a port's performance. Meanwhile, general information on offshore terminal concepts, as well as the operational differences between a land-side terminal and an offshore terminal, are elaborated in Section 3.3. The relation between this research (T9.5) and the previous tasks of the WP9 is illustrated in Figure 3.3. On the right side of the figure, the three levels of planning and decision making in a logistics system are presented.

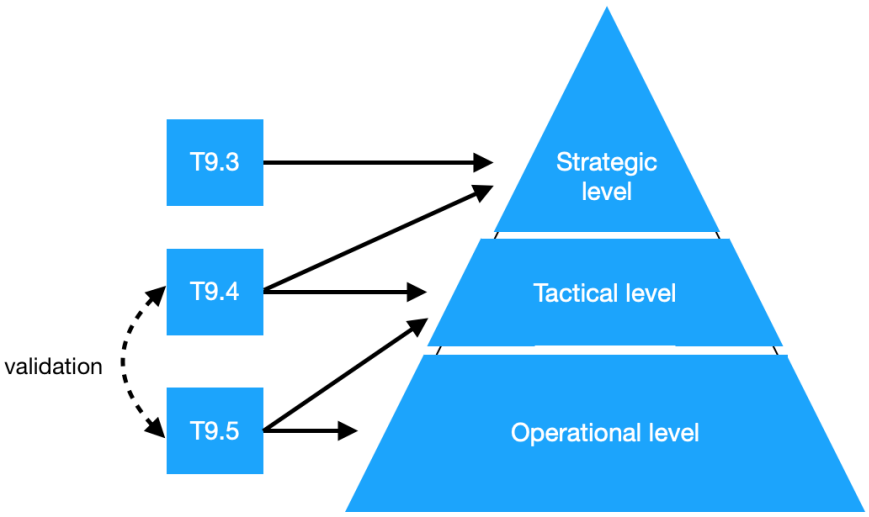


Figure 3.3 Contribution of the WP9 tasks to the three decision-making levels

While T9.3 has introduced some design concepts of the platform, T9.4 has also proposed several configurations for the number of equipment that will be used on the platform. These are strategic and tactical decisions. Meanwhile, this research aims to evaluate the configurations that have been proposed by the previous tasks using a simulation. To validate the simulation model, the result of this research will be compared to the expected performance of the platform which values are obtained from the previous tasks. Furthermore, it is foreseen that the output of this research should be able to support the decision-making process at the operational level.

Chapter 4 Model Development

This chapter elaborates the development of a simulation model that will be used as a tool to answer the main research question. At the end of this chapter, these sub-questions will be answered: *"How should the port and the platform be modelled, and how should the platform coordinate with the port in order to have an integrated container handling service? What are the configurations and scenarios to be evaluated using the model? Does the model behave the way the user intended it to be?"*. Section 4.1 provides an overview of the model, while Section 4.2 elaborates the simulation objects, processes, as well as the interactions between the objects. Section 4.3, Section 4.4, and Section 4.5 describe the construction of different configurations and scenarios. Section 4.6 explains how the model is verified. Lastly, Section 4.7 summarizes the chapter.

The main goal of the model is to evaluate the performance of the Port of Antwerp as well as the S@S platform with respect to different configurations and scenarios. Based on the previous chapters, some criteria have to be fulfilled regarding the model. These criteria are listed below:

a) The model should be able to be used to compare the performance of the port when the platform is installed and not installed,
b) The model should be able to simulate the distribution of containers within the port area as well as between the port and the platform,
c) The model should be able to be used to compare the performance of the platform with different terminal & barge route configurations,
d) The model should be able to generate realistic container handling demand scenarios,
e) The model should be able to incorporate the effect of sea states to the handling capacity of the platform.

4.1 Modelling method

Simulations can either be discrete or continuous, depending on how the system states change. In a container terminal, it is safe to assume that the system states shift at discrete time points. Therefore, the discrete-event simulation method will be used in this research. The simulation model will be an object-oriented model at the container level. This means that each of the port components is modelled as objects with chains of processes that interact with each other.

Nowadays, there is a lot of simulation software with different functionalities and different programming languages. This research chooses to develop a model under the Python programming language for the reason that Python is an emerging, free, and open-source language; which makes it very popular among software developers. There is also a lot of active users, and the user community runs pretty well. When compared to the Pascal programming language used in [24], Python has more learning materials available on the internet.

There are also some options for Python-based discrete event simulation software, e.g. SimPy[1] and Salabim[2]. However, the latest version of SimPy has been modified in some ways that users find it more limiting. Meanwhile, Salabim is a relatively new software with a growing number of users. A comparison of Python/Salabim to Pascal/TOMAS and some other programming languages is included in Appendix G. Accordingly, this research opts for the Salabim software package. The Salabim software is run under JetBrain's PyCharm[3] software as the application development environment.

4.1.1 Model boundaries and assumptions

Boundaries of the model can be depicted as shown in Figure 4.1. In the real world, the container facilities of the port (customs, empty depots, repair centres, etc.) are spread along with the 24 port terminals. However, sea-going container vessels can only be handled at the five existing maritime container terminals. All the containers coming and leaving to/from Port of Antwerp must have been handled at one of these terminals. Moreover, vessels that have smaller call sizes must consolidate their volumes at the consolidation hubs. Several studies [25], [44] have been conducted in order to determine where the consolidation hubs should be located. In these studies, three consolidation hubs are proposed: quay K869, quay K1742, and quay K364. Therefore, this research chooses to include quay K364 as one of the container terminals of the port, and add another terminal as the offshore modular platform.

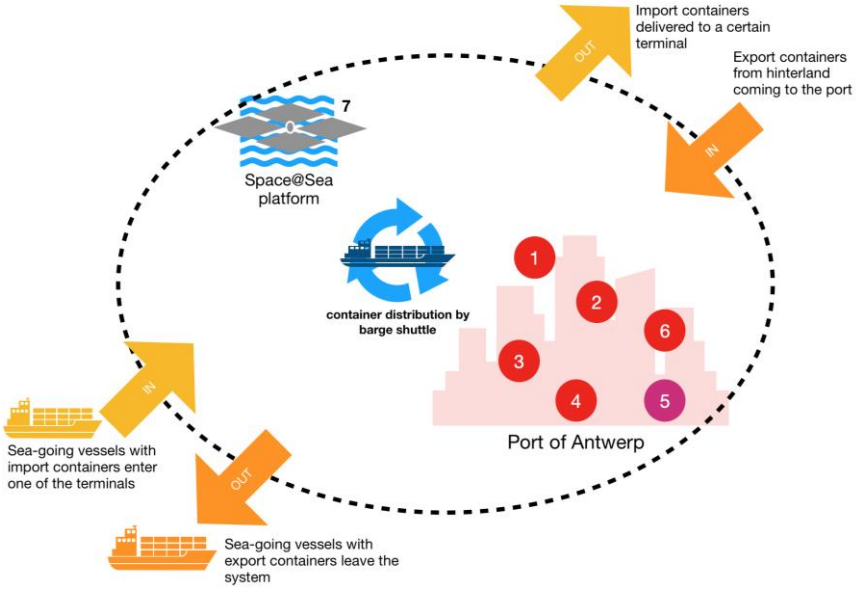


Figure 4.1 Boundaries of the model

In the real world, the distribution of containers around the port area is done with trucks, trains, and the intra-port barge shuttle service. This research assumes that the distribution is done solely by the intra-port barge shuttle service, and fixed sailing times between one terminal to the others. In the simulation, the barges are assumed to operate 24/7, while in the real world the barges only operate 18 hours per day.

The containers are distinguished as import and export containers. The import containers are carried by the sea-going vessels, and they need to be delivered to a certain terminal. Meanwhile, the export containers are the containers coming from the hinterland that need to be loaded onto the vessels. Each container has its due time. Due time of the import containers can be +1, +2, or +3 days after their arrival time. Concurrently, due time of the export containers is the departure time of the vessels they should be loaded onto.

[1] https://simpy.readthedocs.io/en/latest/contents.html/
[2] https://www.salabim.org/
[3] https://www.jetbrains.com/pycharm/download/

Other assumptions regarding the simulation model are listed as follow:

- A vessel visits either the S@S platform or one of the terminals of the Port of Antwerp.
- The S@S platform is assumed to be implemented as an extension for the other terminals in terms of handling containers. Vessels will prefer to call at the port terminals when one of the port terminals is less utilized than the platform.
- There is no direct transhipment service at the platform. In this way, containers that are handled at the platform must go to one of the terminals of the port before being delivered to the hinterland.
- Quay length of the terminals is not taken into account.
- Once a vessel is assigned a certain number of QCs, other QCs that become available from their previous vessel cannot be assigned to the same vessel.
- The barge shuttle service is handled by a dedicated BC in each terminal.
- The barge shuttle's sailing time between the terminals is fixed.
- The traffic of the Scheldt river and wind condition around the port area are considered to be constantly fine.
- All the equipment and the barges operate 24/7 with no downtimes due to maintenance, shift changes, etc.

4.1.2 Model input and output

The input and output of the model are illustrated in Figure 4.2. The input for the simulation includes the terminal configurations, container handling demand scenarios, barge route strategies, and the sea states scenarios. This research refers to the specification of the terminals given in Table 2.2 to determine the number of QCs in each terminal. While the other input is determined from an input file, the container handling demand is generated by sampling from distributions that are implemented as functions in Salabim (see Section 4.3). The examples of the input files are given in Appendix D. Definition of the barge route strategies is given in Section 4.4, and the calculation of factors that are used in the weather input file is given in Section 4.5.

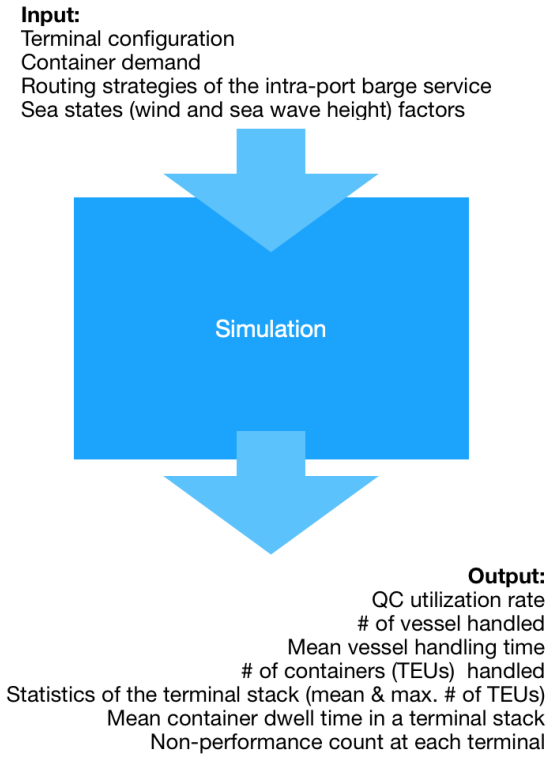


Figure 4.2 'Black box' diagram of the model

On the other hand, the output of the simulation is the performance indicators of the system. These indicators are as follow:

- *QC utilization rate of the terminal* — The ratio of busy QCs to the total number of QCs in the terminals
- *Vessel handled* — The number of vessels that are handled at the terminals
- *Vessel handling time* — The time a vessel spends in a Terminal
- *Containers handled* — The number of containers, in TEUs, handled at a terminal
- *Non-performance count in each terminal* — When containers are delivered later than their due times, they are registered as non-performance
- *Statistics of each terminal stack* — Mean and maximum number of TEUs in each terminal stack
- *Mean dwell time of the containers* — The amount of time containers spends in a stack

Salabim provides a monitor function which allows these indicators to be tracked during the simulation. The statistics of the monitored parameter can be saved to a .txt output file, while the data of the monitored parameters can also be extracted as a .csv file. In this way, the data can be further represented as graphs and charts by using numeric Python software packages, such as *pandas*[4], *numpy*[5], *seaborn*[6], or *matplotlib*[7]. A screenshot of the .txt output file is shown in Figure D.4 of Appendix D.

4.2 Objects of the simulation model

The objects in the simulation model can either be active or passive. They can also be implemented as a control for the other objects. Passive objects do not have any processes embedded in them. Each of the objects has certain attributes and a dedicated function.

The model that will be developed for this research consists of these following objects: a VesselGenerator, Vessels, Containers, Barge, Terminals, QuayCranes, BargeCranes, and a Weather. These objects are listed in Table 4.1 along with their states, and functions.

Table 4.1 List of simulation objects

Object name	State	Function
VesselGenerator	*Active*	*Generates Vessels*
Vessel	*Active*	*Creates Containers and sails to a Terminal*
Container	*Passive*	*The main flowing entity of the model*
Terminal	*Passive*	*Destination points of the Containers*

[4] https://pandas.pydata.org/
[5] http://www.numpy.org/
[6] https://seaborn.pydata.org/
[7] https://matplotlib.org/

Barge	*Active*	*Delivers Containers to the Terminals*
QuayCrane	*Active*	*Unload/Load Containers from Vessels*
BargeCrane	*Active*	*Unload/Load Containers from Barges*
Weather	*Active*	*Generates weather factors*

Schematic of the model is shown in Figure 4.3 below. The following sections will elaborate on each of these objects along with their attributes and processes.

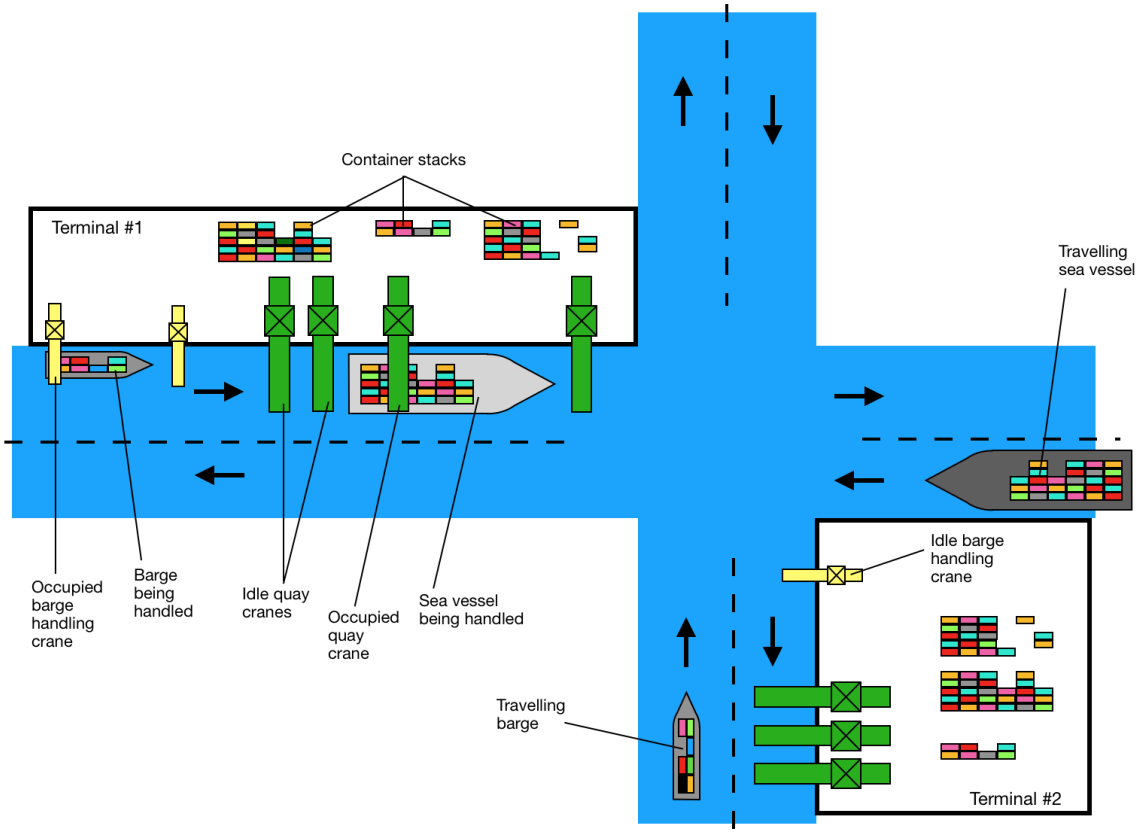


Figure 4.3 Schematic diagram of the model

4.2.1 VesselGenerator

The VesselGenerator creates the Vessels to be handled at the port and the platform. The time between each vessel generation is called the *interarrival_time*.

Attributes:

- *interarrival_time* the time interval between the Vessel's generation

Process description language (PDL):

- *Repeat:*
  - *Create Vessel*
  - *Hold for interarrival_time*

4.2.2 Vessel

When they are created by the VesselGenerator, the Vessels are immediately assigned with their *capacity*, *call size*, *arrival_time*, and *departure_time*.

The *capacity* determines the *max_quaycranes* that can be assigned to the Vessel. This value is determined based on three parameters: a) the vessel length, b) the number of container rows on the vessel, and c) the specification of the QCs. Table 4.2 shows the *max_quaycranes* for each vessel class.

Table 4.2 Relation of Vessel's capacity and max_quaycranes [7], [45]

Vessel class based on size & capacity	Max. number of QCs
≤ 800 TEUs	3
800 TEUs ≤ x ≤ 2500 TEUs	5
2500 TEUs ≤ x ≤ 4500 TEUs	7
4500 TEUs ≤ x ≤ 8000 TEUs	9
≥ 8000 TEUs	12

The *call size* is a term introduced as the total number of Containers that are associated with the Vessel:

$$Vessel's\ call\ size = import\_containers + export\_containers \quad (1)$$

The attribute *arrival_time* is calculated based on the Vessel's creation time:

$$arrival\_time = creation\ time + n_{days}\left(60\frac{minutes}{hour} \times 24\frac{hours}{day}\right) \quad (2)$$

The variable n denotes the number of day between the Vessel's creation time and its arrival time. This can be 1, 2, or 3 days. It also determines the amount of time between when the export Containers arrive at their origin Terminals and when their Vessel has arrive at a certain Terminal. Therefore, it is possible to experiment with different values of n to see how the performance of the system would be affected by this variable.

Meanwhile, the *departure_time* is calculated based on the expected handling time of a Vessel, which is shown below:

$$departure\_time = arrival\_time + (QuayCrane's\ move\_time \times Vessel's\ call\ size) \quad (3)$$

The first task of the Vessels is to wait to be allocated to a certain Terminal. The allocation algorithm is given in the upcoming paragraph. Then, the Vessels create the Containers that need to be (un)loaded from/to themselves. After that, it holds until its arrival time, sails to one of the Terminals and requests for QuayCranes to handle its load. While there are no QuayCranes available, the Vessel will wait until either a) QuayCranes become available, or b) its departure time.

Vessel allocation algorithm

In this research, an algorithm is proposed to allocate the Vessels to the terminals. The algorithm is based on three parameters: a) the Vessel's capacity, b) the storage capacity of the S@S platform, and c) the Terminal's available QuayCrane.

As the Vessels have a random *arrival_time,* and they do not enter the Terminal immediately after their creation, it is not possible to decide in which Terminal the Vessel should call to based on the current utilization rate of the Terminal. Therefore, a simple QC and capacity booking system are implemented in the algorithm by introducing three attributes for the Terminals.

The first attribute is the Terminal's *booked_quaycrane,* which is used to predict the Terminal's available QuayCranes at the Vessel's *arrival_time*. The initial value of this variable is zero. Every time a Vessel is allocated to a certain Terminal, it books some QuayCranes based on its *max_quaycrane* and updates the *booked_quaycrane* value. When the Vessel reaches the destination Terminal, it unbooks the QuayCranes and again updates the value. The equation used is shown below:

$$Prediction\ for\ available\ QC\ in\ the\ Terminal = length\ of\ Terminal's\ quaycraneQ - Terminal's\ booked\_quaycrane \quad (4)$$

As mentioned before, this research assumes that the S@S platform is implemented as an extension to the port terminals in terms of handling containers. Vessels will prefer to call to the port terminals if they are available.

The second and third attributes are specifically made for the S@S platform. These attributes are *booked_capacity* and *max_storage_capacity*. While the value of the *booked_capacity* is updated every time by the Vessel's process, the *max_storage_capacity* remains constant.

The initial value of the *booked_capacity* is set to zero. After the destination has been decided, the Vessel updates the *booked_capacity* by adding its *export_containers*. This will allow the export Containers to be stored on the S@S platform until the Vessel's *arrival_time*. When the Vessel arrives at the S@S platform, it again updates the value by detracting the *export_containers*. Meanwhile, the import Containers of the Vessel are taken into account in this booking system by including the length of the S@S platform stack in the decision equation.

The algorithm is shown in Figure 4.4. Each of the limiting parameters is user-configurable. For instance, a user of the model could set the maximum/minimum Vessel size that could be handled at the platform, or specific maximum storage capacity for the S@S platform.

Attributes:

- *capacity* — the size of the Vessel
- *max_quaycranes* — the maximum number of QuayCranes that can be assigned to the Vessel
- *export_containers* — the number of Containers to be loaded onto the Vessel
- *import_containers* — the number of Containers to be unloaded from the Vessel
- *arrival_time* — the timestamp when the Vessel should arrive at a Terminal
- *departure_time* — the timestamp when the Vessel should leave the Terminal. This is used in the non-performance rate calculation
- *destination* — an integer with a value between 1 to 6 corresponding to the Terminals' IDs. It is sampled from a uniform distribution.
- *import_load* — a queue for Containers to be unloaded from the Vessel
- *export_load* — a queue for Containers to be loaded onto the Vessel

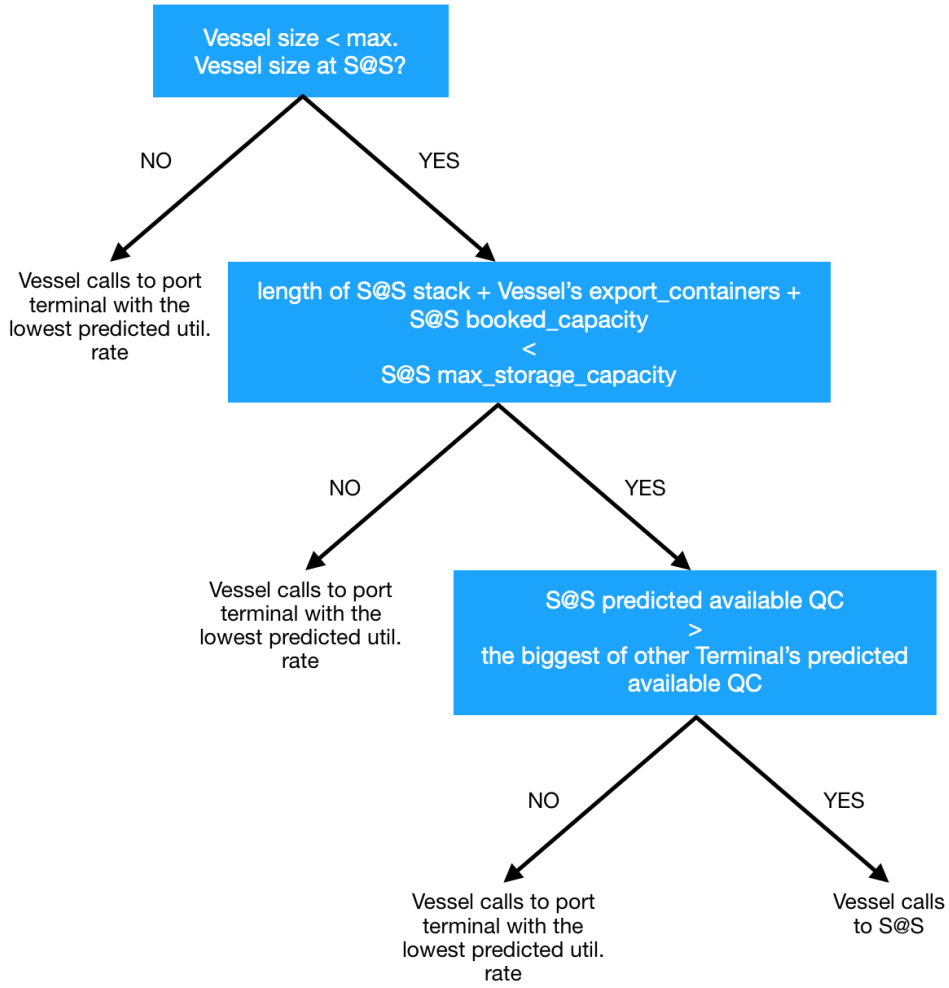


Figure 4.4 Vessel allocation algorithm

Process description language (PDL):

- *Wait to be allocated to a Terminal (see Figure 4.4)*
- *Book QuayCrane*
- *If destination = SpaceAtSeaPlatform:*
  - *Book storage capacity*
- *Create the Containers to be unloaded and assign attributes*
- *Create the Containers to be loaded and assign attributes*
- *Hold until arrival_time*
- *Unbook QuayCrane*
- *If destination = SpaceAtSeaPlatform:*
  - *Unbook storage capacity*
- *Enter the queue of Vessels at the destination Terminal*
- *While there is no available QuayCrane at the Terminal:*
  - *wait until there is available QuayCrane or until departure_time*
- *If there is available QuayCrane at the Terminal:*
  - *Request QuayCrane*
- *Else leave Terminal and register import_load as non-performance*

4.2.3 Container

The Containers are the main flowing entity in the model. They are passive objects, so they have no process assigned to them. There are two types of Container: export and import. The

export Containers are generated in a Terminal stack, while the import Containers are generated as the load of a Vessel. All the attributes of the Containers are assigned by the Vessels when the Containers are generated.

The Containers are created in the Vessels' process to ensure that each of them is linked to a certain Vessel. In this way, their creation time will always be ahead of the Vessel's *arrival_time*. Thus, as explained in Section 4.2.2, the Vessel's attribute *arrival_time* determines the amount of time an export Container has to arrive at a certain Terminal in prior to its Vessel's arrival to one of the Terminals.

Attributes:

- *vessel* — the Vessel on which the Container must be (un)loaded.
- *due_time* — the latest time a Container must be either delivered to the destination Terminal or loaded onto a Vessel. This attribute is used in the non-performance rate calculation.
- *origin* — an integer between 0 and 6 that denotes the Terminal ID where the Container is originated. If the Container is an import container, the value is 0. If the Container is an export container, the value is a sample of a uniform distribution between 1 to 6.
- *destination* — an integer between 1 and 6 that denotes the Terminal ID where import Containers need to be delivered, or where the export Containers need to be loaded onto their Vessel. If the Container is an import container, the value is a sample of a uniform distribution between 1 to 6. If the Container is an export container, the value is equal to the Vessel's attribute *destination*.

The attribute *due_time* is calculated based on one of the Vessel's attributes; either the *arrival_time* or the *departure_time*. If the Container is an import container, the *due_time* is calculated as below:

$$due\_time = Vessel's\ arrival\_time + n_{days}\left(60\frac{minutes}{hour} \times 24\frac{hours}{day}\right) \quad (5)$$

The value of n can be 1, 2 or 3 days. In contrast, if the Container is an export container, the *due_time* is calculated with the following equation:

$$due\_time = Vessel's\ departure\_time \quad (6)$$

Register non-performance

The 'Register non-performance' method is called by either the Vessel, the QuayCranes or the BargeCranes. This method checks if the Containers are delivered before their due times. The process can be described as follow:

- *If Container's due_time is < now:*
  - *non_performance_count + 1*

The non-performance rate of the whole system can be calculated with the following equation:

$$Non - Performance\ Rate = \frac{non\_performance\_count}{containers\_generated} \times 100\% \quad (7)$$

The *containers_generated* is a variable that is counted every time Containers are created. Meanwhile, the non-performance rate of each terminal is calculated with the following equation:

$$Terminal's\ share\ to\ the\ non-performance\ rate = \frac{late\_delivery}{containers\_handled} \times 100\% \quad (8)$$

Both the *late_delivery* and *containers_handled* are attributes of the Terminals, which corresponds to the number of Containers that are delivered later than their due time in the Terminal, and the number of Container entries to the Terminal's *stack*, respectively.

4.2.4 Terminal

The Terminal is where the Vessels are handled and where the Containers are (un)loaded from/to Vessels and Barges by the QuayCranes and BargeCranes. It is a passive object which also acts as a destination point for the import Containers. Each of the Terminal has a *stack* where the Containers are put into while waiting to be further handled. The numbers of QCs and BCs in a Terminal are determined from the input file *'Terminal[ID#].txt'*. The input file for the S@S platform is named as *'SpaceatSeaPlatform.txt'*. Screenshot of the input file is given in Figure D.1. The S@S platform is set to have two other attributes, namely the *booked_capacity* and *max_storage_capacity*. These attributes are used in the Vessel's decision algorithm (see Section 4.2.2).

Attributes:

- *specification* — an attribute assigned to the input file *'Terminal[ID#].txt'*
- *id* — ranges from 1 to 7
- *quaycraneQ* — a queue for QuayCranes. The length of this queue is monitored as a performance indicator 'utilization rate of the terminal'
- *vesselQ* — a queue for Vessels while they are being handled or waiting for available QuayCrane
- *bargeQ* — a queue for Barges while they moor at the Terminal
- *stack* — a queue where the Containers wait to be further handled. The export Containers are generated in this queue. The length of stay in this queue is implemented as performance indicator 'mean dwell time of the containers'
- *booked_capacity* — A variable used in the Vessel allocation algorithm to decide whether to call at S@S platform or the other Terminals
- *max_storage_capacity* — A variable used in the Vessel allocation algorithm to decide whether to call at S@S platform or the other Terminals
- *containers_handled* — The number of Container entries to the *stack* queue
- *late_delivery* — The number of Containers that are delivered later than their *due_time*

4.2.5 Barge

The Barge sails between the Terminals to deliver Containers to their destinations. When it arrives at a Terminal, it requests an available BargeCrane and moors at the Terminal for its *mooring_time*. After that, the Barge leaves the Terminal and sail to the next Terminal for its *sailing_time*. The route and sailing time between the Terminal as well as the mooring time for each Terminal can be modified by altering the input file *'case[ID#].txt'*. A screenshot of the input file is given in Figure D.2.

Attributes:

- *my_route* — an attribute assigned to describe the input file for the Barge
- *terminal* — the Terminal where the Barge is mooring at
- *mooring_time* — determines the amount of time the Barge spent at a specific Terminal. The *mooring_time* for each Terminal can be modified by altering the input file *'case[ID#].txt'*

- *sailing_time* determines the amount of time the Barge spent to sail to the next Terminal. The *sailing_time* for each Terminal can be modified by altering the input file *'case[ID#].txt'*
- *load* a queue for the Containers that are loaded onto the Barge

Process description language (PDL):

- *Repeat:*
  - *Read my_route to determine the next Terminal*
  - *Enter the Terminal*
  - *Activate an idle BargeCrane in terminal*
  - *Read my_route to determine the mooring_time*
  - *Hold for mooring_time*
  - *Passivate the corresponding BargeCrane*
  - *Unassign the current Terminal (terminal = None)*
  - *Read my_route to determine the sailing_time*
  - *If current terminal OR next terminal is SpaceAtSeaPlatform:*
    - *Hold for sailing_time * Weather factor*
  - *Else hold for sailing_time*

4.2.6 QuayCrane

The QuayCrane unloads Containers from a Vessel, then loads Containers to a Vessel. The time required for both loading and unloading is called the *move_time*. After finished handling a Vessel, or if the Vessel has reached its *departure_time*, the QuayCrane checks for late Containers in its Terminal *stack* and register these Containers as non-performance.

Attributes:

- *move_time* determines the amount of time needed to load/unload a Container to/from a Vessel
- *vessel* the Vessel it is currently handling
- *terminal* the Terminal where it belongs

Process description language (PDL):

- *Repeat:*
  - *Leave Terminal's quaycraneQ*
  - *Unload Containers from vessel*
  - *If Vessel's import_load /= 0:*
    - *Register Vessel's import_load as non-performance*
  - *Load Containers to vessel*
  - *Register late Containers in Terminal's stack as non-performance*

The unloading process can be described in the PDL below:

- *While Vessel's import_load /= 0 and Vessel's departure_time < now:*
  - *Unload a Container from myvessel's importload*
  - *If terminal = SpaceAtSeaPlatform:*
    - *Hold for move_time * Weather factor*
  - *Else hold for move_time*

Finally, all the Containers that need to be loaded on the corresponding Vessel and available in the Terminal will be loaded onto the Vessel before the Vessel's *departure_time*, and this process is elaborated in the following PDL:

- *While Vessel's export_load /= Vessel's export_containers and Vessel's departure_time < now:*

            - *Select the first Container in terminal's stack that needs to be loaded to vessel*
            - *Load Container to vessel*
            - *If terminal = SpaceAtSeaPlatform:*
                - *Hold for move_time * Weather factor*
            - *Else hold for move_time*

4.2.7 BargeCrane

The BargeCrane handles the Barges in the Terminal where it is located. It unloads Containers from the Barge and calls for the non-performance registration. Then, it loads Containers to the Barge in a *while loop*. It is activated and passivated by the Barges.

Attributes:

- *move_time* — determines the amount of time needed to load/unload a Container to/from a Barge
- *barge* — the Barge it is currently handling
- *terminal* — the Terminal where it belongs

Process description language (PDL):

- *Repeat:*
    - *For Containers in barge's load with destination = terminal's id:*
        - *If Container is import Container:*
            - *Register Container as non-performance*
        - *Elif Container is export Container, put Container in terminal's stack*
    - *While there are no Containers in terminal's stack to be loaded onto the Barge, standby*
    - *For Container in terminal stack:*
        - *If Container's destination ≠ terminal's id, load Container onto the Barge*
        - *If terminal = SpaceAtSeaPlatform:*
            - *Hold for move_time * Weather factor*
        - *Else hold for move_time*

4.2.8 Weather

The Weather is a non-physical object responsible for the generation of the sea states factors. These factors are used in the calculation of the *move_time* of QCs and BCs as well as *sailing_time* of the Barges.

The Weather generates two factors for the S@S platform. These factors are the wind and the sea wave height efficiency factors. Each factor ranges from 0 to 1. The values are determined from an input file 'inputweather.txt.' A screenshot of the weather input file is given in Figure D.3. A thorough discussion of the sea states scenario is given in Section 4.5.

Attributes:

- *wind_factor* — A value between 0 and 1 which represents the wind condition at the platform
- *seawave_factor* — A value between 0 and 1 which represents the wave condition at the platform
- *timestep* — The amount of time between updates of the factors. This is set to 1 hour (60 minutes).

Process description language (PDL):

- *Repeat:*
    - *Read input file for Weather condition*

- *Assign values to each of the attributes that represent the weather condition*
- *Hold for timestep*

4.3 Container handling demand distributions

The vessels and containers generation rates are defined based on a historical dataset of container handling demand at the Port of Antwerp. The original dataset has been analyzed and cleaned from so it can be used as a starting point for tasks T9.4 and T9.5 of the WP9 Transport&Logistics@Sea. The analysis of this dataset is included in Appendix C.

4.3.1 Vessel's inter-arrival time

The total number of 20-ft and 40-ft containers in the historical dataset corresponds with the total number of containers handled at the Port of Antwerp in the year 2017, which is about 6.5 million containers. This is equivalent to 10.5 million TEUs [4]. Therefore, it is assumed that this dataset represents the demand for the year 2017. The historical dataset consists of 5656 rows, each representing a vessel. Therefore, the mean interarrival time of the vessels, in minutes, can be calculated with the equation below:

$$inter\ arrival\ time = \frac{365 \times 24 \times 60}{5656} = 92.1\ minutes \tag{9}$$

In the simulation model, the interarrival time of the VesselGenerator is sampled from a normal distribution with specifications stated below:

$$VesselGenerator's\ interarrival\ time = Normal\ Distribution\ N(\mu, \sigma^2) \tag{10}$$

$$mean\ (\mu) = 92\ minutes \tag{11}$$

$$standard\ deviation\ (\sigma^2) = 20\ minutes \tag{12}$$

4.3.2 Vessel capacity distribution

Distribution of the vessel capacity visiting the Port of Antwerp is given in Appendix C. From this distribution, a cumulative distribution is made. The distribution is listed in Table 4.3. This cumulative distribution is used by the VesselGenerator to sample each of the Vessel's capacity.

Table 4.3 Vessel's capacity cumulative distribution

Vessel's capacity	Cumulative distribution (%)
≤ 100 TEUs	0
≤ 1,500 TEUs	36.4
≤ 3,000 TEUs	53.4
≤ 4,500 TEUs	66
≤ 6,000 TEUs	76.1
≤ 10,500 TEUs	90.8
≤ 21,000 TEUs	100

4.3.3 Vessel call size distribution

The number of import and export containers are sampled from cumulative distributions that are made based on Appendix C. These distributions are shown in Table 4.4.

Table 4.4 Call size cumulative distributions

Number of TEUs	Export cum. distribution (%)	Import cum. distribution (%)
≤ 100 TEUs	0	0
≤ 500 TEUs	30	30
≤ 1,000 TEUs	60	60
≤ 1,500 TEUs	75	80
≤ 2,000 TEUs	85	90
≤ 2,500 TEUs	90	95
≤ 3,000 TEUs	95	98
≤ 6,000 TEUs	100	100

4.4 Barge route strategies

When the S@S platform is not present, the barge shuttle only sails between the terminals of the port. This is shown in Figure 4.5. Meanwhile, there are two barge route strategies between the port and the platform that will be evaluated in this research. These strategies are shown in Table 4.5. In the first strategy, the barge shuttle sails between the terminals and the S@S platform. This strategy is shown in Figure 4.6. In the second strategy, which is illustrated in Figure 4.7, there are two barge routes. The first one is a loop route within the port area, while the second one is a connection between the port and the platform. The mooring and sailing durations between the terminals are determined from the actual schedule of the PBS [16]. In this simulation, the barge is assumed to operate 24 hours a day; despite the fact that in the real world the barge only operates 18 hours a day.

Table 4.5 Barge route strategies

Strategy no.	Barge route
1	Loop route
2	Loop route within the port area and a connection between the port and the platform

Although this research only looks at two strategies, the model can be used to evaluate other kinds of barge route strategy. This can be done by creating more barges or adjusting the input file for the Barge *'case[ID#].txt'*. Screenshot of this input file is shown in Figure D.2.

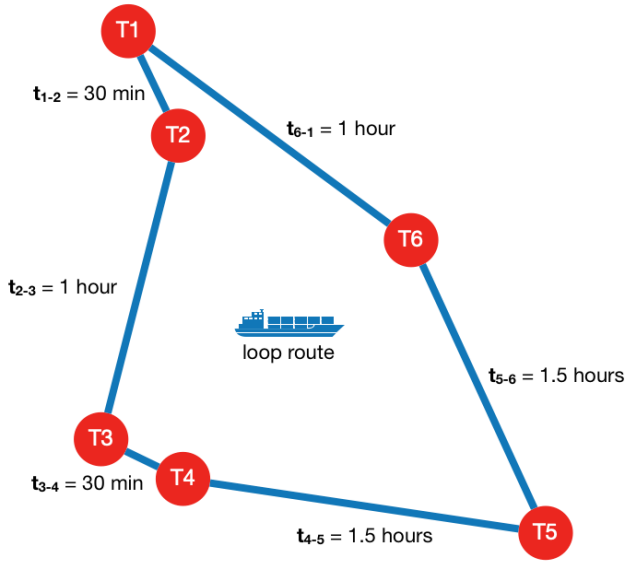


Figure 4.5 Loop route strategy without S@S platform

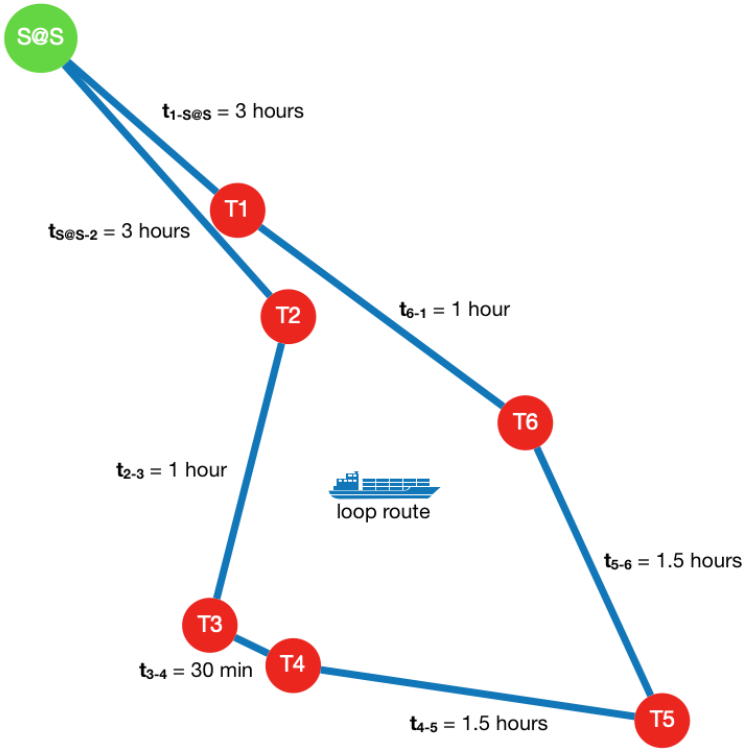


Figure 4.6 Loop route strategy with S@S platform

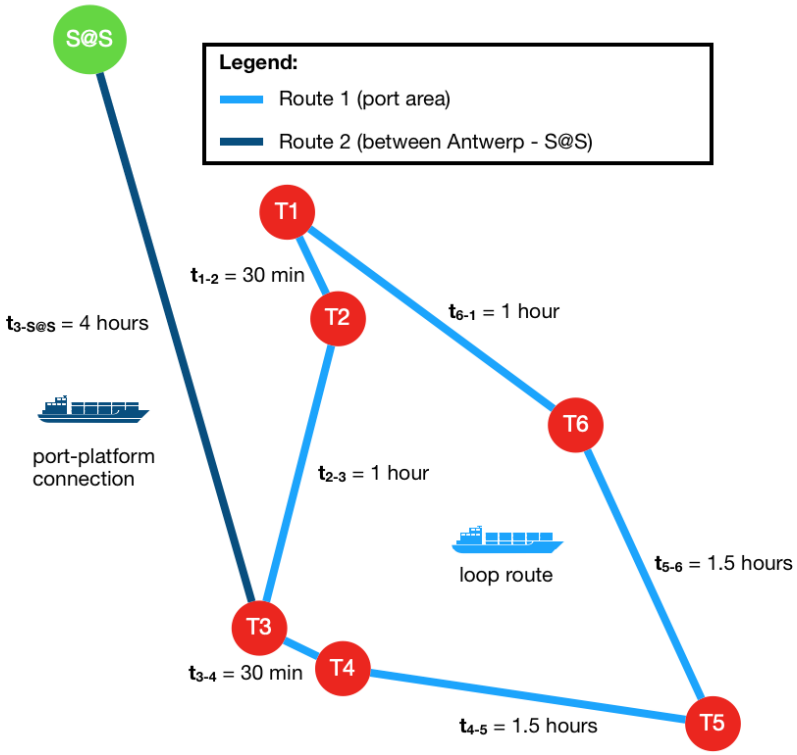


Figure 4.7 Two barge routes strategy

4.5 Sea states scenario

In this section, a simple sea states scenario is constructed based on the wind speed and significant sea wave height at the location of the platform. It is expected that the other S@S project partners will provide a more realistic scenario based on the same dataset that is used in this research.

To take into account the effect of these sea states into the calculation of the QuayCrane's and BargeCrane's *move_time* as well as the Barge's *sailing_time*, mathematical relations between these sea states to the handling capacity of the equipment, as well as the sailing time of the barges, are required. These relations are the *wind and sea wave efficiency factors*. The efficiencies are used in the calculation of *move_time* of the QCs and BCs at the S@S platform:

$$move\_time\ at\ S@S = \frac{move\_time}{wind\_factor \times seawave\_factor} \tag{13}$$

Meanwhile, if one Barge sails from the port to the platform or vice versa, the *sailing_time* will be calculated with the equation below:

$$sailing\_time_{PoA-S@S} = \frac{sailing\_time}{seawave\_factor} \tag{14}$$

The wind and sea wave factors in the model are constructed based on the historical dataset at the location of the platform. The dataset is available in the MetOcean DataPortal[8]. In order to reduce the number of variables, the values in the dataset are categorized. This categorization is shown in Table 4.6.

[8] https://www.metocean-ondemand.com/

Table 4.6 Categorization of the sea states [31], [35], [36]

Wind speed categorization (@ 10m above ground)	**Efficiency factor (%)**
0 m/s < v_{wind} ≤ 10 m/s	100
10 m/s < v_{wind} ≤ 20 m/s	75
20 m/s < v_{wind} ≤ 30 m/s	50
Sea wave height categorization	**Efficiency factor (%)**
0 m < $h_{sea\text{-}wave}$ ≤ 1.5 m	100
1.5 m < $h_{sea\text{-}wave}$ ≤ 3 m	50
$h_{sea\text{-}wave}$ > 3 m	25

4.6 Verification of the model

The verification of a model checks if the model behaves as it is specified. This step determines if the model is well-implemented and could be used to evaluate the performance of the real-scale system. The behaviour of each simulation object can be checked by using both the trace monitor and animation function that is provided in Salabim.

4.6.1 Trace monitor and animation window

Salabim allows traces of the simulation to be monitored. The trace can be printed to the monitor of the PyCharm IDE by toggling *trace=True* in the main code. In this way, the monitor is updated with the current process in the simulation. The monitor is shown in Figure 4.8. At the most left column, a user can see the current simulation time. Moving to the right is the current object that is doing a certain process. Next to it is the current process in the simulation, while the most right shows an annotation regarding the current object or process.

```
431                              bargecrane.13 hold              scheduled for   4447.000 @  431+
339+   4446.969 quaycrane.20     current
341                              container.764                   leave vessel
339                              quaycrane.20 hold               scheduled for   4447.969 @  339+
337+   4447.000 quaycrane.82     current
341                              container.126                   leave vessel
350                              container.126                   enter export stack
351                              unloaded container trigger      value = True --> False allow inf components
337                              quaycrane.82 hold               scheduled for   4448.000 @  337+
433+   4447.000 bargecrane.7      current
437                              container.11030                 leave export stack
438                              container.11030                 enter barge
443                              bargecrane.7                    enter barge crane Q
414                              bargecrane.7                    leave barge crane Q
433                              bargecrane.7 hold               scheduled for   4448.000 @  433+
431+   4447.000 bargecrane.13     current
437                              container.36                    leave export stack
438                              container.36                    enter barge
443                              bargecrane.13                   enter barge crane Q
414                              bargecrane.13                   leave barge crane Q
431                              bargecrane.13 hold              scheduled for   4448.000 @  431+
339+   4447.969 quaycrane.20     current
341                              container.765                   leave vessel
350                              container.765                   enter export stack
351                              unloaded container trigger      value = True --> False allow inf components
339                              quaycrane.20 hold               scheduled for   4448.969 @  339+
337+   4448.000 quaycrane.82     current
341                              container.127                   leave vessel
350                              container.127                   enter export stack
351                              unloaded container trigger      value = True --> False allow inf components
337                              quaycrane.82 hold               scheduled for   4449.000 @  337+
433+   4448.000 bargecrane.7      current
437                              container.11033                 leave export stack
438                              container.11033                 enter barge
443                              bargecrane.7                    enter barge crane Q
414                              bargecrane.7                    leave barge crane Q
433                              bargecrane.7 hold               scheduled for   4449.000 @  433+
431+   4448.000 bargecrane.13     current
437                              container.37                    leave export stack
438                              container.37                    enter barge
443                              bargecrane.13                   enter barge crane Q
414                              bargecrane.13                   leave barge crane Q
431                              bargecrane.13 hold              scheduled for   4449.000 @  431+
339+   4448.969 quaycrane.20     current
```

Figure 4.8 Trace monitor of Salabim in PyCharm

The simulation processes can also be visualized with the animation window function that is provided in Salabim. The animation function can show objects that are in queues and real-

time performance of the model. An animation window has been made during the model development so that every essential process can be indicated from it. The animation window shows the terminals of the port and the S@S platform as white nodes in a network that are connected with blue lines. This is illustrated in Figure 4.9. At the middle of the window, a user can see the locations of the barges as well as the length of their loads. The non-performance rate can also be found under the barge status. Each of the nodes are connected to their indicators. The indicators correspond with the performance indicators of the model. The weather factors at the platform are also shown at the right of the S@S platform indicators.

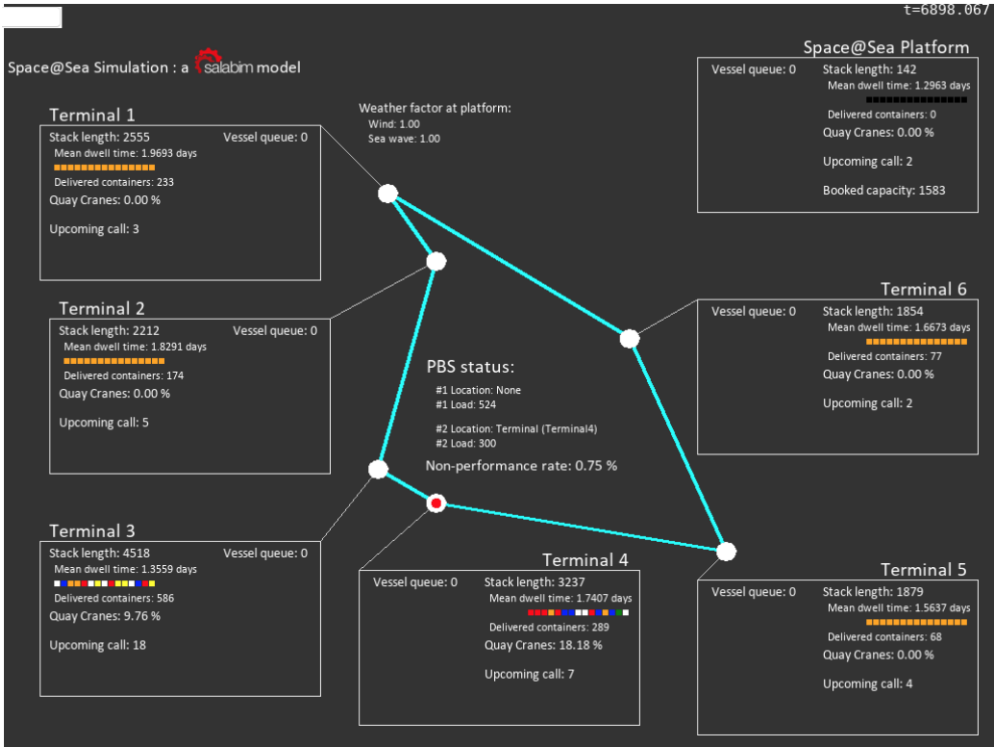


Figure 4.9 Simulation animation window

Together with the trace monitoring function, this animation window can be used to verify the behaviour of the model by giving a visualization of the simulation. Some of the checks that are conducted during this verification phase are discussed in the following paragraphs.

4.6.2 Test run configuration

In order to check the behaviour of the model, some test runs are conducted. These runs have a different configuration from the real problem configuration, which is intendedly made so that the problem becomes simpler and the capabilities of the model could be further examined. Unless stated otherwise, most of the test runs are run under the configuration listed below:

Vessel & Container demand:

- Vessel capacity cum. distribution — 75% ≤ 1,000 TEUs
100% ≤ 2,000 TEUs
- Vessel call size — Export: 50% capacity
Import: 50% capacity
- Vessel arrival time — 1 to 3 days from creation, uniformly distributed
- Due time of the import containers — 1 to 3 days from vessel arrival, uniformly distributed
- Expected handling time per container — 0.5 minutes

Terminal configuration:

- Number of terminals — 3 + 1 S@S platform
- Number of QC per terminal — Terminal 1: 5 QCs
Terminal 2: 10 QCs
Terminal 3: 15 QCs
S@S platform: 5 QCs
- Maximum QC per vessel — 5 (fixed for all vessel size)
- Max. vessel capacity at S@S platform — 1,000 TEUs
- Max. storage capacity at S@S platform — 2,000 TEUs

Barge routing strategy:

- Number of barges — 2
- Barge #1 route — Terminal 1 – Terminal 2 – Terminal 3
  - Sailing time — 60 minutes between the terminals
  - Mooring time — 60 minutes at each terminal
- Barge #2 route — Terminal 3 – S@S platform
  - Sailing time — 120 minutes between the terminals
  - Mooring time — 120 minutes at each terminal

4.6.3 Demand generation check

A *print trap* is implemented in the code so that every time a Vessel is created the monitor would show the Vessel capacity and call size. It has been checked during the simulation that the Vessel's capacity and the call sizes matches the pre-determined distributions.

The Containers have different destinations and are either import or export. The destination is checked by assigning colours to the Containers. The colour specification is given in Table 4.7.

Table 4.7 Colours assigned to the containers based on their destinations

Destinations	Colours
Terminal 1	Blue
Terminal 2	Red
Terminal 3	Green
Terminal 4	Yellow
Terminal 5	White
Terminal 6	Orange
SpaceAtSeaPlatform	Black

To conduct this check, the simulation is run for a long time with only one Vessel generated along the simulation time. When the Vessel is created, there should be a group of Containers with the same colour entering all Terminal stacks. These are the export Containers that needs to be loaded onto the Vessel. At the beginning of the simulation run, the sum of all Containers inside the stacks, which are shown per terminal in the animation window, should correspond with the number of export Containers associated with the first generated Vessel. The same

colour indicates that these Containers need to be transported to a Terminal where the Vessel they are associated with will make its call. Meanwhile, as it is assumed that the S@S platform is not connected directly to the hinterland, there will be no export Containers generated from the S@S platform. The import Containers will start to show up in the animation window at the arrival time of the Vessel. As these Containers have different destinations, their colour should vary. All these conditions have been verified by using both the trace monitor and the animation window.

4.6.4 Vessel allocation algorithm check

The Vessel has an algorithm which represents the allocation of a certain Vessel to one of the Terminals based on the Terminal's availability. This algorithm has three different parameters that are user-configurable: a) the vessel capacity constraint, b) the maximum storage capacity at the platform, and c) comparison of the utilization rate of the terminals.

With the test configuration shown in Section 4.6.2, it is possible to predict the destination of the Vessels. The first three Vessels should call at either Terminal 3 or Terminal 2, regardless of their capacity. If the fourth Vessel has a capacity ≤ 1,000 TEUs, it should call at S@S platform. Otherwise, the fourth Vessel will call at Terminal 1. If this is the case, the next Vessel with capacity ≤ 1,000 TEUs will call at the S@S platform.

Meanwhile, if the fourth Vessel calls at S@S platform, the 5th, 6th, and 7th Vessel will call at Terminal 1, Terminal 2, and Terminal 3. Then, if the 8th Vessel has a capacity ≤ 1,000 TEUs, it will call at S@S platform. If this is the case, then all the storage capacity of the S@S platform has been booked, and the remaining Vessels will call at either Terminal 1, Terminal 2, or Terminal 3, depending on the utilization rate of these terminals.

However, if the 8th Vessel has a larger capacity, it will call at Terminal 1 and neglect the available storage capacity at the S@S platform. If this is the case, then as soon as a Vessel with capacity ≤ 1,000 TEUs is generated, it will call at the S@S platform and claim this remaining storage.

The conditions above are checked by monitoring the values of 'upcoming call' indicators that are shown in the animation window, and the model behaves accordingly.

4.6.5 QuayCrane operation check

In order to check the operation of the QuayCranes, a test run is conducted with slight modifications in the configuration. The first modification is that there is no Barge present in the system, so it is impossible to distribute the Containers among the Terminals. The second one is that Vessels can only call at Terminal 1. In this way, there will be a concentrated Vessel queue at Terminal 1 waiting to be handled by the QuayCranes. Other modifications of the configuration are also listed below:

- Vessel interarrival time — 50 minutes
- Vessel capacity distribution — 100% 1,000 TEUs, export:import=1:1
- Maximum QC per vessel — 3 QCs

The first Vessel that moors at Terminal 1 will request QC1, QC2, and QC3 to handle its load. These three QuayCranes will spend 500 minutes from its arrival time to unload all the import Containers, load the available export Containers, and finally wait for the undistributed Containers. From this time on, a Vessel will arrive at Terminal 1 every 50 minutes.

Meanwhile, upon its arrival, the second Vessel will only be handled by two QuayCranes; QC4 and QC5, as the other three are still busy with the first one. Fifty minutes later, the third Vessel should come to Terminal 1 and wait for any available QuayCrane.

At the time when the first Vessel leaves the Terminal, the third Vessel should immediately claim QC1, QC2, and QC3 to handle its load while QC4 and QC5 should remain to wait for the second Vessel's export Containers. Fifty minutes later, when the second Vessel leaves the Terminal, QC4 and QC5 will be requested by the fourth Vessel to handle its load. At this time, QC1, QC2, and QC3 should still be busy with unloading import Containers from the third Vessel.

All the conditions above have been checked, and the model behaves just as it is specified.

4.6.6 Vessel bailing check

To check the bailing process of a Vessel, a similar test run is conducted with a slight variation in the third Vessel's capacity. Instead of using a sample from distributions, this test run uses a pre-determined text file to assign the call sizes of the Vessels. Then, the third Vessel capacity is set to 500 TEUs instead of 1,000 TEUs. In this way, the third Vessel should bail after waiting for 250 minutes at Terminal 1. This condition is met during the test run, implying that the bailing process is verified.

4.6.7 Barge operation check

The Barge operation can be checked with the animation window. Each of the Terminals is provided with a *bargeQ* in which Barges enter when they reach a Terminal. The queue is animated in such a way so that every time a Barge moors at a Terminal the white node will be filled with either a blue or red circle. The blue circle annotates PBS #1 while the red circle annotates PBS #2. The locations and the number of loads of the Barges are also updated in real time under the 'Barge status' indicators. The BargeCrane waits while there are no Containers on the Barge's *load* or in the Terminal's *stack*. It has been checked that the Barges moor and sail according to the specification given in the input file.

4.6.8 Weather check

One of the indicators shown in the animation window is the 'Weather factors at the platform.' This indicator shows the values of both the wind speed and sea-wave height factors that are used to calculate the QuayCrane's and BargeCrane's *move_time* while handling Containers as well as the Barge's *sailing_time* while travelling to/from the platform. The Weather component is set to update these factors in an hour timestep, so the values that are shown in the animation window should also change every 60 minutes with values that are obtained from the '*inputweather.txt*' file. Furthermore, when the factors are not 1, the QuayCranes at SpaceAtSeaPlatform should move slower, and the Barge that goes from/to the platform should sail longer. These conditions have been checked from the animation window and trace monitor, and the simulation components act as predicted.

4.6.9 Parameter consistency & sensitivity check

The purpose of this check is to see if the model output is either consistent or sensitive to the changes in the model parameter. Therefore, this check focuses on the shift of values in the statistics of the Terminal's *vesselQ* and *stack*, which are included in the output file.

A Vessel is handled by multiple QuayCranes at the same time, and the QuayCranes are set to be always occupied by the Vessel until the Vessel's *departure_time*. In this way, most of the time the import Containers will be completely unloaded from the Vessels before the *departure_time*. However, as the QuayCranes will have to wait for the export Containers to arrive at the Terminal, the amount of export Containers that can be loaded into the same Vessel by these QuayCranes depends more on the barge operation. Therefore, changing the QuayCrane that are assigned to the Vessel, or changing the expected handling time per container (which determines the amount of time a Vessel spends at a certain Terminal), should have more effects on the model than changing the QuayCrane's *move_time*.

In contrast, the BargeCrane's *move_time* should have more significant effects on the model performance. If the BargeCrane's *move_time* is set to a smaller value, there should be more

Containers loaded to the Barges during their call in Terminals and vice versa. By lowering the *move_time*, the distribution of Containers among the Terminals should be done faster, resulting in a higher number of Containers that arrived at their destination on-time. The Barge's *load* would also increase, indicating that the system requires barges with larger capacity. However, at a certain point, the number of Containers inside the Barge's *load* would not be realistic.

Changing the number of Barges in the system should also have a significant impact on the model performance, with the same reasons as explained in the previous paragraph. However, in contrast to changing the Barge's *move_time*, adding more Barge would not increase the Barge's *load*. It is also vital that each Terminal has sufficient BargeCranes in case multiple Barges moor at the same Terminal. Otherwise, one of the Barges would not be handled, which will be indicated by a stagnant period of the Barge's *load* monitor.

4.7 Summary of the chapter

This chapter has provided the answers to the sub-questions that are related to the model development. In Section 4.1, a general description of the simulation model is given. The model is built based on the *Delft System Approach*, in which the input will be processed in some functions along with some pre-determined requirements. The model output is the performance indicators of the port-platform system. The objects and processes of the simulation are discussed in Section 4.2. A vessel allocation algorithm is proposed to distribute the vessel calls among the terminals and the platform. The full code of the model is given in Appendix E. The container handling demand scenarios are made based on a historical dataset of the vessels visiting the Port of Antwerp in year 2017. There are two different barge route configuration which will be evaluated with the model. The effect of the wind speed and sea waves to the productivity of the terminal equipment as well as the sailing time of the barge shuttle are taken into account in the simulation model. All of these are explained in Section 4.3, Section 4.4, and Section 4.5, respectively. Lastly, some checks are conducted using the model to see if the model behaves as described in the process description. This is elaborated in Section 4.6. The trace monitor and animation window are used to verify the behaviour of the model. The consistency and sensitivity of the configurable parameters in the model are also checked by observing the output files. It can be concluded that the model behaves as it is specified.

Chapter 5 Experiments & Results

This chapter will answer the last group of sub-questions: *"Does the model represent the behaviour of the actual system? If so, how does the platform affect the performance of the port when different configurations and scenarios are implemented?"*. Section 5.1 describes how an experimental plan is made to evaluate the different configurations and scenarios regarding the port-platform logistics operations. Section 5.2 examines if the output of the simulation model represents the performance of the real-world system. Section 5.3 to 5.8 explore the performance differences between the different configuration and scenarios based on the simulation results. Lastly, Section 5.9 summarizes the chapter.

5.1 Experimental plan

In order to investigate how the performance of the port is affected by the presence of the S@S platform, several cases are constructed with different barge route configurations and vessel size preferences. These cases are shown in Table 5.1.

Table 5.1 Experiment plan

Case #	Barge route strategy	Vessel allocation strategy
1 (ref.)	(S@S platform not present)	-
2	A loop route	≤ 6,000 TEUs
3	A loop route	≥ 6,000 TEUs
4	Two different routes	≤ 6,000 TEUs
5	Two different routes	≥ 6,000 TEUs

Each of the cases is simulated for 60,480 minutes (6 weeks) in the simulation world. This is equivalent to 2-4 hours in real time; depending on the complexity of the configurations and scenarios.

Case #1 will show the performance of the system when the S@S platform is not present. The result of case #1 is compared to available historical data and result of similar studies in order to validate the model. In case #2-#5, the S@S platform is present. The result of these cases will be compared to those of case #1 to see how the port's handling capacity is affected by the presence of the platform with different configurations.

Apart from what is listed in Table 5.1 and explained in the previous paragraphs, these other settings are shared among the five cases:

- Due time of the import containers — 1 to 3 days from vessel arrival, uniformly distributed

- Expected handling time per container — 0.5 minute
- Number of terminals — 6 + 1 S@S platform
- Number of QC per terminal — Terminal 1: 12 QCs
Terminal 2: 8 QCs
Terminal 3: 41 QCs
Terminal 4: 11 QCs
Terminal 5: 5 QCs
Terminal 6: 5 QCs
S@S platform: 10 QCs
- Max. storage capacity at S@S platform — 10,000 TEUs

Furthermore, a preferred strategy is selected from these five initial cases for each of the barge route strategy and vessel allocation strategy. The chosen strategies are used to conduct sensitivity analysis to provide additional insights on how the S@S platform would perform with respect to other configurations and scenarios, such as: a) the number of QCs on the S@S platform, b) the location of the S@S platform, c) different distributions of the vessel's inter-arrival time and call sizes, and d) different sea states scenarios.

5.2 Validation of the model

The validation phase of a model checks if the model is suitable to represent the real problem. A model would never be completely the same as the real system. Therefore, this section describes to what extent that the model can be described as 'valid'.

In case #1-#5, the interarrival time is sampled from a non-negative normal distribution $N(\mu, \sigma^2)$ with mean $\mu = 100\ minutes$ and standard deviation $\sigma^2 = 20\ minutes$ (see Section 4.3.1). Therefore, the number of vessels that are generated within this simulation period can be predicted with the equation below:

$$prediction\ of\ \#\ of\ vessels = \frac{60480\ minutes}{100\ {}^{min.}/_{vessel}} = 605\ vessels \tag{15}$$

Instead, the simulation results show that there are 612 vessels generated in each simulation. Meanwhile, there are 942,826 TEUs of containers that are generated and associated with these vessels. This value is lower than the predicted number of containers that should enter the system within the 6-week period, which is calculated based on the number of TEUs handled at the Port of Antwerp in the year 2017 [4], [46]:

$$prediction\ of\ \#\ of\ TEUs = 10.5\ million\ TEUs \times \frac{6\ weeks}{1\ year\ (52\ weeks)} = 1.2\ million\ TEUs \tag{16}$$

Nevertheless, the demand differences between the simulation results and the predicted values are considered to be valid, given that they are still within the tolerable area of the pre-determined distributions.

Table 5.2 shows the simulation results of the five cases. The result of case #1 is expected to replicate the performance of the real port when the platform is not installed. It is observed from the result that the overall mean QC utilization rate of the port is 47.5%, which is lower than the expected value (70-90%). This is caused by the assumption that QCs that have finished handling a vessel cannot be reallocated to another vessel that has started to be handled by other QCs. Therefore, in the simulation, it is possible that a vessel is only handled by two QCs even though there are other available QCs in the terminal. Despite, this value corresponds with a report which stated that the utilization rate of the Port of Antwerp is relatively low compared to those of the other ports [47]. At the same time, the values for the vessel handling time is always lower than one day, and this corresponds with the value mentioned in [48].

Table 5.2 Results of the five case simulations

	#1: S@S not present	#2: A loop route, S@S handles vessels ≤ 6,000 TEUs	#3: A loop route, S@S handles vessels ≥ 6,000 TEUs	#4: Two barge routes, S@S handles vessels ≤ 6,000 TEUs	#5: Two barge routes, S@S handles vessels ≥ 6,000 TEUs
Port Performance					
Mean QC utilization rate of the port terminals:	47.38%	45.59% *(-1.79%)*	47.33% *(-0.06%)*	42.65% *(-4.73%)*	41.52% *(-5.86%)*
Mean vessel handling time at port terminals (h):	10.46	12.16 *(+14%)*	11.83 *(+13.08%)*	11.17 *(+6.82%)*	9.82 *(-6.13%)*
Max. # of TEUs at the port terminals in avg. (TEUs):	12708	13686 *(+7.70%)*	13285 *(+4.55%)*	12588 *(-0.94%)*	11907 *(-6.30%)*
Mean container dwell time at port terminals (days):	1.66	2.02 *(+21.41%)*	1.97 *(+18.76%)*	1.71 *(+2.85%)*	1.68 *(+0.88%)*
Platform Performance					
Mean QC utilization rate:	-	47.43%	54.98%	38.14%	52.02%
Vessel handled:	-	87 *(14.2%)*	45 *(7.35%)*	89 *(14.54%)*	46 *(7.51%)*
Mean vessel handling time (h):	-	11.35	17.05	9.22	18.41
Containers handled (TEUs):	-	87464 *(9.28%)*	72482 *(7.69%)*	85188 *(9.03%)*	74121 *(7.86%)*
Mean # of TEUs in stack (TEUs):	-	1856	1337	1099	951
Max. # of TEUs in stack (TEUs):	-	6319	6404	5423	4979
Mean container dwell time (days):	-	0.884	0.771	0.531	0.517
Non-performance rate	-	2.98%	1.90%	1.11%	2.34%

As mentioned earlier in Section 2.1, the Port of Antwerp claims an overall QC productivity of 40 crane moves per hour per crane. This means that in the real world a QC takes about 1-2 minutes to load/unload a container to/from a vessel. However, if the QuayCrane's and BargeCrane's *move_time* in the model are set to 1.5 minutes, the non-performance rate of the model will reach about 70-80% of the total containers that are generated during the simulation time. This high value of non-performance rate is not only caused by the *move_time* values but also due to two other reasons: a) the assumption that the container distribution is done solely by the barge shuttle service, and b) the vessels and containers are handled only based on *first in, first out* (FIFO) algorithm. In order to have a lower non-performance rate, there needs to be a dedicated planning and control scheme to allocate vessels based on their departure times and to sort containers based on their due times. However, this planning and control scheme is out of the scope of this research. Therefore, in order to compensate the need for the planning and control scheme, this research recommends setting the QuayCrane's and BargeCrane's *move_time* to lower values. Though, it is important to note that lowering the QuayCrane's and BargeCrane's *move_time* would also affect the statistics of the terminal stacks. If the *move_time* value is too low, the containers will leave the terminal stack too quickly, and the number of containers that are stored in the terminals and their dwell time values would become unrealistic.

It is also possible to compensate the high non-performance rate by adding more barge shuttles which operate only in the port area. In this way, the container distribution between the port terminals will be enhanced, and the non-performance error will be localized to the S@S platform. Though, this non-performance localization is not relevant for case #2 and case #3, due to the implemented barge route strategy. The effect of adding more barge shuttles in the port area is illustrated in Figure 5.1. It is seen from the figure that the non-performance rate of case #1 reaches a value close to zero if there are 12 barge shuttles operating in the

port area. With the same amount of barge shuttles that operate in the port area, the non-performance rate of case #4 decreases from about 55% to 5%. However, adding more barge shuttles will significantly increase the computational load of the simulation.

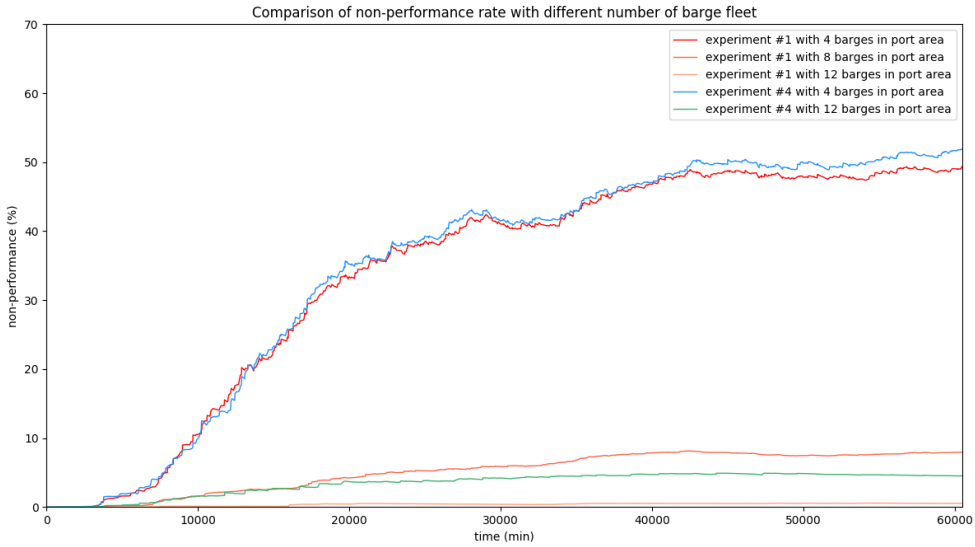


Figure 5.1 Comparison of non-performance rate with different number of barge shuttles

For these reasons, this research chooses to compensate the high non-performance rate value by setting the QuayCrane's *move_time* to 0.5 minute, and the BargeCrane's *move_time* to 0.1 minute. The QuayCrane's *move_time* is equivalent to a QC productivity rate of 120 TEUs/hour. On the other hand, the low value of the BargeCrane's *move_time* can be considered as having four BCs at the terminals to handle the barge shuttle at once. Though these values seems too optimistic, they provide a trade-off point between the high non-performance rate and the validity of the statistics of the terminal stack. With these configurations, the statistics of the terminal stacks in case #2 and #4 can be compared to the Space@Sea memorandum document. This document is included in Appendix C. In this document, the dwell time of containers in the port terminals is assumed to be between 3-5 days, while the dwell time on the platform is assumed to be 1-3 days. These assumptions are used to calculate predictions of the platform's storage capacity with respect to different demand scenarios. In the simulations, though, the container dwell time is not an assumption, but a measure of the port performance. The results of the simulations show that the mean container dwell time at the port terminal ranges between 1.5-2 days, while the mean container dwell time on the platform ranges between 0.5-1 days. Despite the difference, both the memorandum document and the simulation results show that the container dwell time on the platform is always half of the dwell time at the port terminal. Moreover, with the same demand scenario, it is also observed from the simulation results that the maximum number of containers on the platform is also half of the memorandum's predicted storage capacity of the platform.

5.3 Comparing port-platform performance with different strategies

Even though there are 612 vessels that are generated within the 6-week period, only 577 vessels have arrived and handled at one of the terminals. In addition, the number of containers that have been handled by the system ranges between 850,000-900,000 TEUs. It can be seen from Table 5.2 that the installation of the S@S platform will allocate some of the vessels to be handled at the platform. When the platform is set to handle vessels with capacity ≤ 6,000 TEUs (case #2 and #4), around 15% of the total vessels are handled at the platform. When the platform is set to handle vessels with larger capacity (case #3 and #5), only 7.5-8% of the total vessels are handled at the platform. Concurrently, the installation of the S@S platform will also allocate some of the containers to be handled at the platform. The percentages of containers that are handled at the S@S platform ranges from 7.5-9.5%. Case

#2 has the highest percentage (9.28%), followed by case #5 (7.86%). The percentages of case #3 and #4 are 7.35% and 9.03%, respectively. The statistics of the S@S platform stack (mean, maximum, and dwell time of the containers) reach the highest values in case #2 and the lowest values in case #5.

Figure 5.2 shows the comparison of the port's performance in the five simulation cases. In general, the installation of the S@S platform reduces the average QC utilization rate of the port terminals. In case #2 and #3, the maximum number of containers in the port terminals increases by 7.7% and 4.5%, respectively; while in case #4 and #5, this number decreases by 0.94% and 6.3%, respectively. Therefore, it can be concluded that the installation of the S@S platform with the two-barge-route strategy will reduce the port's QC utilization rate and required storage capacity.

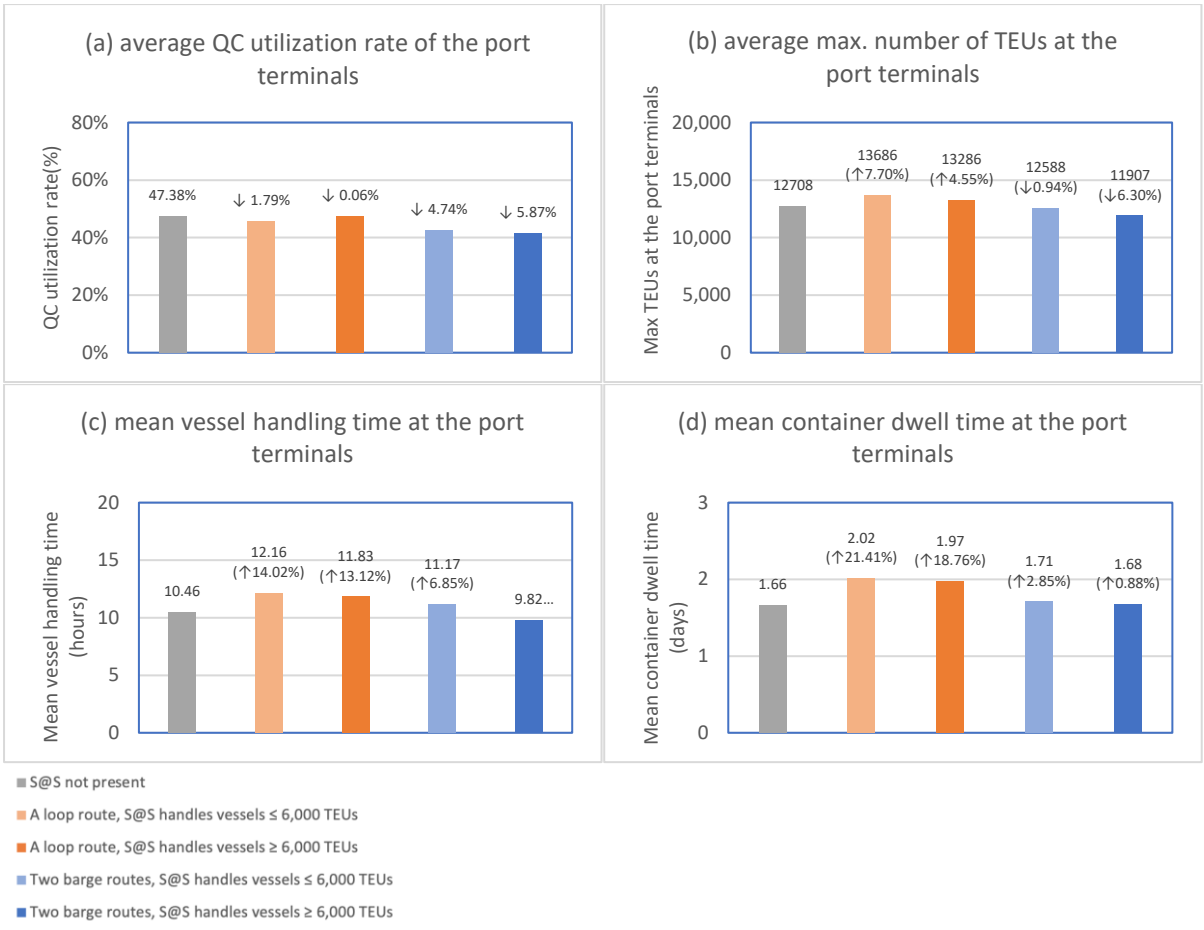


Figure 5.2 Comparison of the port's performance with different strategies

Except for case #5, the mean vessel handling time at the port terminals always increase when the S@S platform is present. The increases become more significant when the S@S platform only handle vessels ≤ 6,000 TEUs. This is because a portion of the small vessels is handled at the S@S platform. Therefore, the average vessel size that visits the port terminal increases. Larger vessels tend to have larger call sizes. The vessels have pre-determined departure time which is based on their call sizes, so it is likely for the large vessels to stay longer at the terminals while they are being handled. However, this parameter is also affected by the container distribution service, as vessels are set to be able to leave before their pre-determined departure time if all their containers have been handled. This is observed in case #3: even though the large vessels are handled at the S@S platform, the mean vessel handling time at the port terminals still increases. This is due to the low distribution service in case #3. Meanwhile, the mean container dwell time always increase when the S@S platform is present; because more export containers have to wait in terminal #3 before being transferred to the S@S platform.

Figure 5.3 shows the comparison of the S@S platform performance with different strategies. In general, with the same vessel allocation strategy, the two barge routes configuration provides lower QC utilization rate than the single loop route strategy. Moreover, with the same barge route strategy, the platform's QC utilization rate is always lower when the platform only handles vessels ≤ 6,000 TEUs. The mean vessel handling time at the platform shows a similar trend with the QC utilization rate.

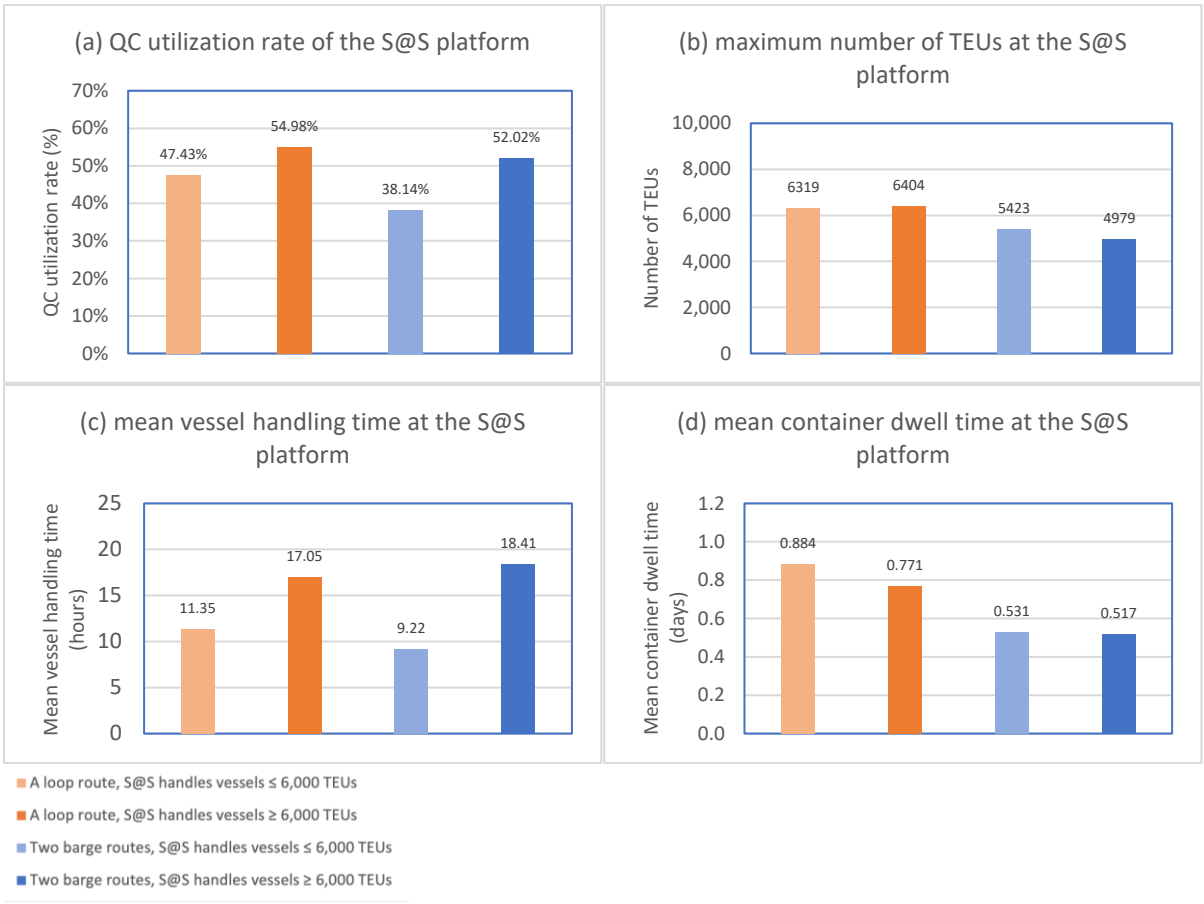


Figure 5.3 Comparison of the platform's performance with different strategies

The maximum number of containers that are stored on the S@S platform, as well as the mean container dwell time, are higher in case #2 and #3 than in case #4 and #5. This is due to the service rate of the barge shuttle. In case #2 and #3, the barge shuttle visits the platform once in a day; while in case #4 and #5 the barge shuttle visits the platform multiple times in a day. Therefore, in case #2 and #3, the containers have to wait longer before being transferred to the port terminals.

The main purpose of the S@S platform is to reallocate some of the handling demand at the port to the platform. It is preferred that the platform can handle more demand with less QC and storage space utilization. Also, even though the S@S platform installation increases the overall container handling time of the port, it is still preferred that the containers that are handled at the platform should be delivered on time. In this case, a lower non-performance rate is preferred. Therefore, this research suggests implementing the strategies from case #4 for the following reasons:

- It provides the highest percentage of vessels that are handled at the platform (14.54%) and the second-highest percentage of containers that are handled at the platform (9.03%). The highest container percentage is in case #2. However, due to the lower container distribution service rate, case #2 has a higher non-performance share of the S@S platform when compared to case #4.
- As shown in Figure 5.3, case #4 provides a better S@S platform performance. Case #5 might have a lower maximum number and dwell time of containers that are stored on

the platform, but it also has a lower percentage of containers that are handled at the platform.

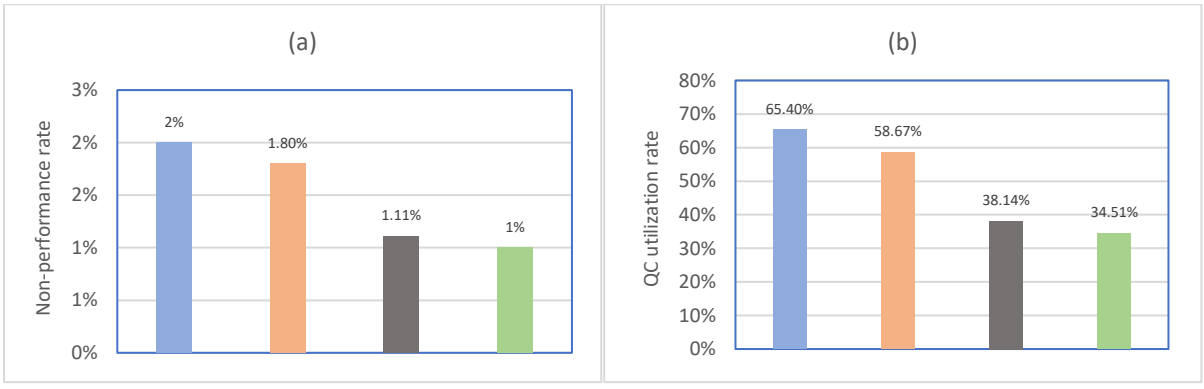
To have more insights on the platform's performance with respect to other configurations, the following sections will discuss several sensitivity analyses that are conducted using the strategies from case #4.

5.4 Comparing different number of quay cranes at the platform

Four simulations are conducted with different number of QCs at the S@S platform. The results are shown in Table 5.3. The 10-crane configuration is used as a reference. Even when the mean QC utilization rate decreases by 3.63%, the 15-crane configuration increases the percentage of vessels and containers that are handled at the platform by 2.5% and 2% respectively. Though not significant, the mean vessel handling time also increases. This might be because more vessels are handled at the platform. The reductions in the S@S share to the non-performance and the maximum number of containers that are stored on the S@S platform indicate that adding more QCs allows more export containers to be loaded onto their vessels on time. Comparison of the platform's performance with different number of QCs is illustrated in Figure 5.4.

Table 5.3 Results of different number of QCs at the platform

Platform performance	**Number of QC on the platform**			
	2 QC	**5 QC**	**10 QC (ref.)**	**15 QC**
Non-performance:	2% *(+0.89%)*	1.8% *(+0.69%)*	1.11%	1% *(-0.11%)*
Mean QC utilization rate:	65.40% *(+27.26%)*	58.67% *(+20.53%)*	38.14%	34.51% *(-3.63%)*
Vessel handled:	72 *(-2.5%)*	76 *(-2%)*	89	101 *(+2.5%)*
Mean vessel handling time (h):	12.73 *(+38.07%)*	11.74 *(+27.33%)*	9.22	9.38 *(+1.74%)*
Containers handled (TEUs):	78281 *(-0.73%)*	79519 *(-0.6%)*	85188	100157 *(+2%)*
Mean # of TEUs in stack (TEUs):	*903 (-17.83%)*	933 *(-15.10%)*	1099	1298 *(+15.33%)*
Max. # of TEUs in stack (TEUs):	3272 *(-39.66%)*	4819 *(-11.13%)*	5423	4588 *(-15.39%)*
Mean container dwell time (days):	0.483 *(-9.03%)*	0.485 *(-8.66%)*	0.531	0.539 *(+1.51%)*



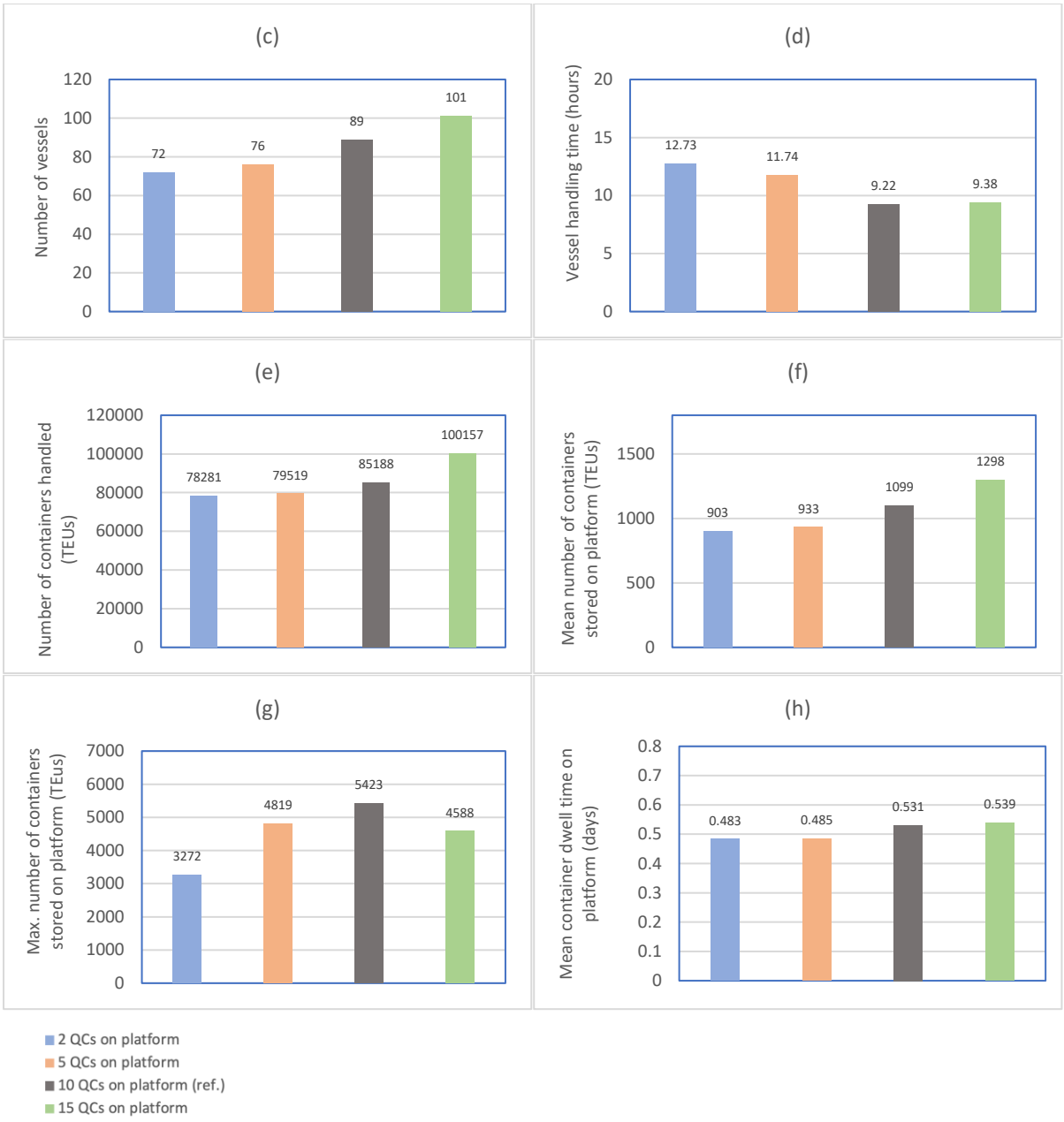


Figure 5.4 Comparison of the platform's performance with different number of QC on the platform

In contrast, the percentage of vessels that are handled at the platform decreases by 2% in the 5-crane configuration and by 2.5% in the 2-crane configuration. The percentage of containers that are handled at the platform decrease by 0.6% in the 5-crane configuration and by 0.73% in the 2-crane configuration. At the same time, the QC utilization rate increases by 20.53% in the 5-crane configuration and by 27.26% in the 2-crane configuration. Moreover, reducing the number of QCs at the platform increases the non-performance rate by about 1%, and the mean vessel handling time at the platform by 2.5-3.5 hours. Meanwhile, the mean container dwell time does not change significantly with the different QC configuration.

5.5 Comparing different platform locations

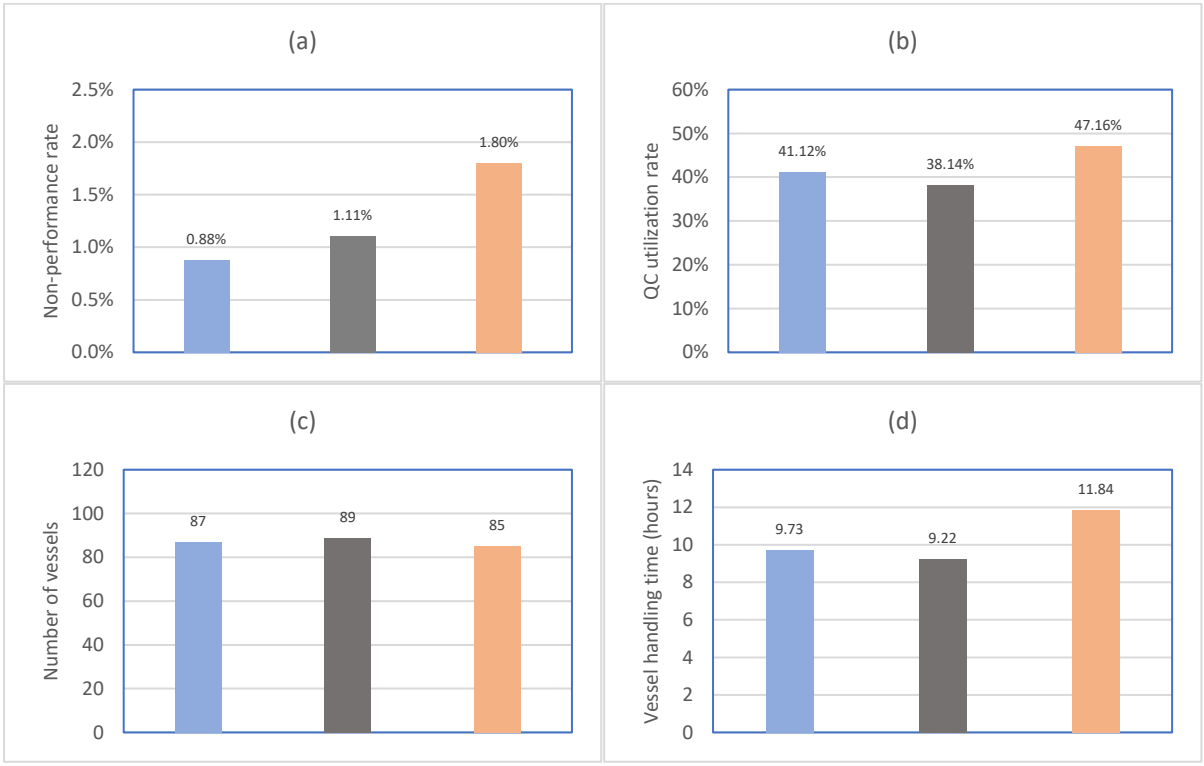
Apart from the pre-determined location (see Figure 1.1), two other potential locations of the platform are examined by altering the sailing time between the port and the platform. The first potential location is the mouth of the Scheldt river. It is assumed that the sailing time between the port and this particular location is 2 hours. The other location is the border of

the offshore operations, at which it is assumed that the barge shuttle should sail for 6 hours from the port to reach the platform.

The results of these simulations are shown in Table 5.4. It is observed from the table that closer location reduces the S@S platform's share to the non-performance as well as the maximum number and dwell time of containers that are stored on the platform; and vice versa. However, the changes are not significant. Figure 5.5 shows the comparison of the platform's performance with respect to the different platform locations.

Table 5.4 Results of different locations of the S@S platform

Platform performance	Sailing time to platform		
	2 hours	4 hours (ref.)	6 hours
Non-performance:	0.88% *(-0.23%)*	1.11%	1.8% *(+0.69%)*
Mean QC utilization rate:	41.12% *(+2.98%)*	38.14%	47.16% *(+9.02%)*
Vessel handled:	87 *(-0.33%)*	89	85 *(-0.65%)*
Mean vessel handling time (h):	9.73 *(+5.53%)*	9.22	11.84 *(+28.42%)*
Containers handled (TEUs):	86467 *(+0.14%)*	85188	99871 *(+1.56%)*
Mean # of TEUs in stack (TEUs):	1035 *(-5.82%)*	1099	1446 *(+31.57%)*
Max. # of TEUs in stack (TEUs):	5200 *(-4.11%)*	5423	5951 *(+9.74%)*
Mean container dwell time (days):	0.495 *(-6.78%)*	0.531	0.602 *(+13.37%)*



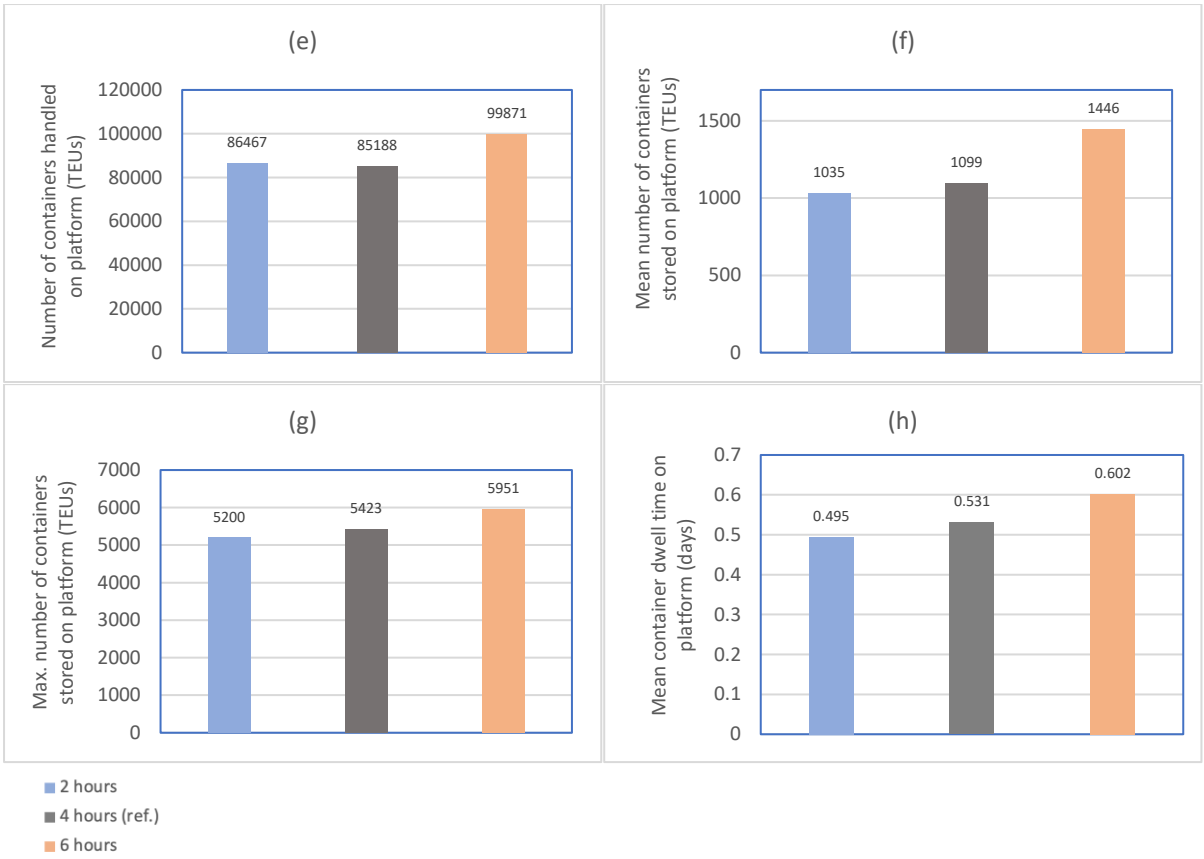


Figure 5.5 Comparison of the platform's performance with different sailing time to platform

Though the location of the platform is not taken into account in the vessel allocation algorithm, the different number of vessels that are handled at the platform in each simulation indicates that there is an effect of the platform location to the allocation of the vessels. The platform location directly affects the barge operation, which plays an important role in the container distribution between the port and the platform. At the same time, the container distribution affects the QC occupancy and availability at the terminals. Therefore, the platform location indirectly affects the QC occupancy and availability, which are taken into account in the allocation algorithm. Due to the same reason, there are also no trends observed in the mean QC utilization rate, mean vessel handling time, as well as the number of containers that are handled at the platform.

5.6 Comparing different demand scenarios

In the future, the demand for container handling at the Port of Antwerp will increase. Therefore, this research will make an assumption in order to construct some future container handling demand scenarios. The Antwerp Port Authority does not include any demand projection in their annual publication, so this research will use the same projections as the ITT studies for Maasvlakte 1&2. There are four future economic trends considered in the ITT studies. These trends are: a) *Low Growth*, b) *European Trend*, c) *Global Economy*, and d) *High Oil Price*. As presented in Table 5.5, each of these trends predicts the number of containers to be handled at the Port of Rotterdam in the year 2030. The demand growth factors are the ratio of the baseline (the year 2008) to the respective future trend. The Global Economy trend has the highest factor, so this research will use the value 2.76.

Table 5.5 Future demand projections for the year 2030 [21]

Projections	Demand in Rotterdam (x10⁶ tonnes/year)	Demand growth factor
Baseline (2008)	112.30	1.00
2030 Low-Growth	190.00	1.69
2030 European Trend	267.00	2.38
2030 Global Economy	310.00	**2.76 (chosen)**
2030 High Oil Price	218.00	1.94

The comparison of container demand in Antwerp and Rotterdam is illustrated in Figure 5.6. From this figure, one can safely assume that the demand growth factor from the year 2008 to the year 2017 is 1.15. Based on this assumption, the growth factor from the year 2017 to 2030 can be calculated as follow:

$$growth\ factor_{2017-2030} = \frac{1}{1.15} \times 2.76 = 2.4 \tag{17}$$

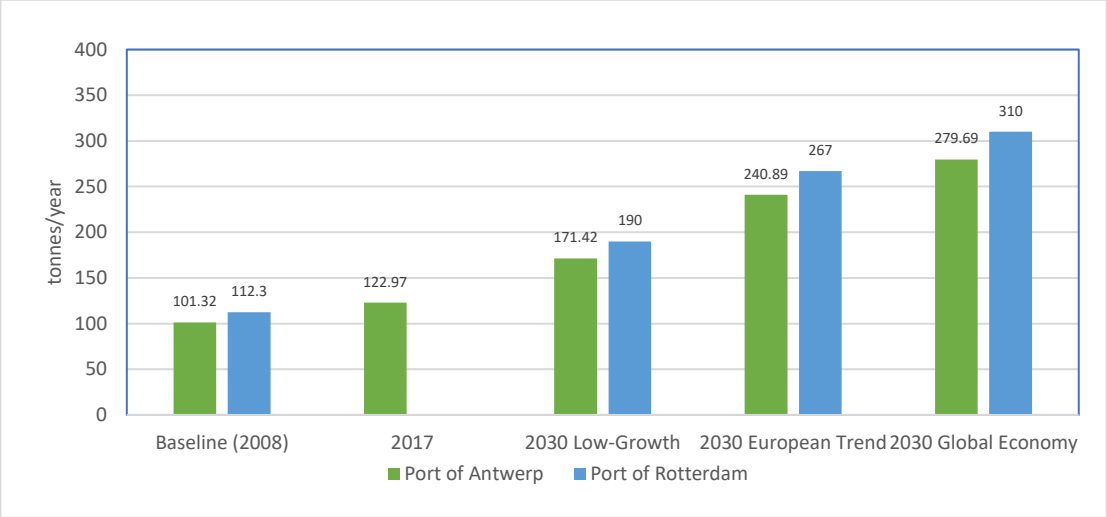


Figure 5.6 Comparison of container handling demand in Antwerp and Rotterdam [4], [21]

This value is used to construct several future demand scenarios that are evaluated with the simulation model. The demand scenarios include the following: a) increase of the vessel's interarrival time, b) increase of the vessel's call sizes.

5.6.1 Increase of vessels' interarrival time

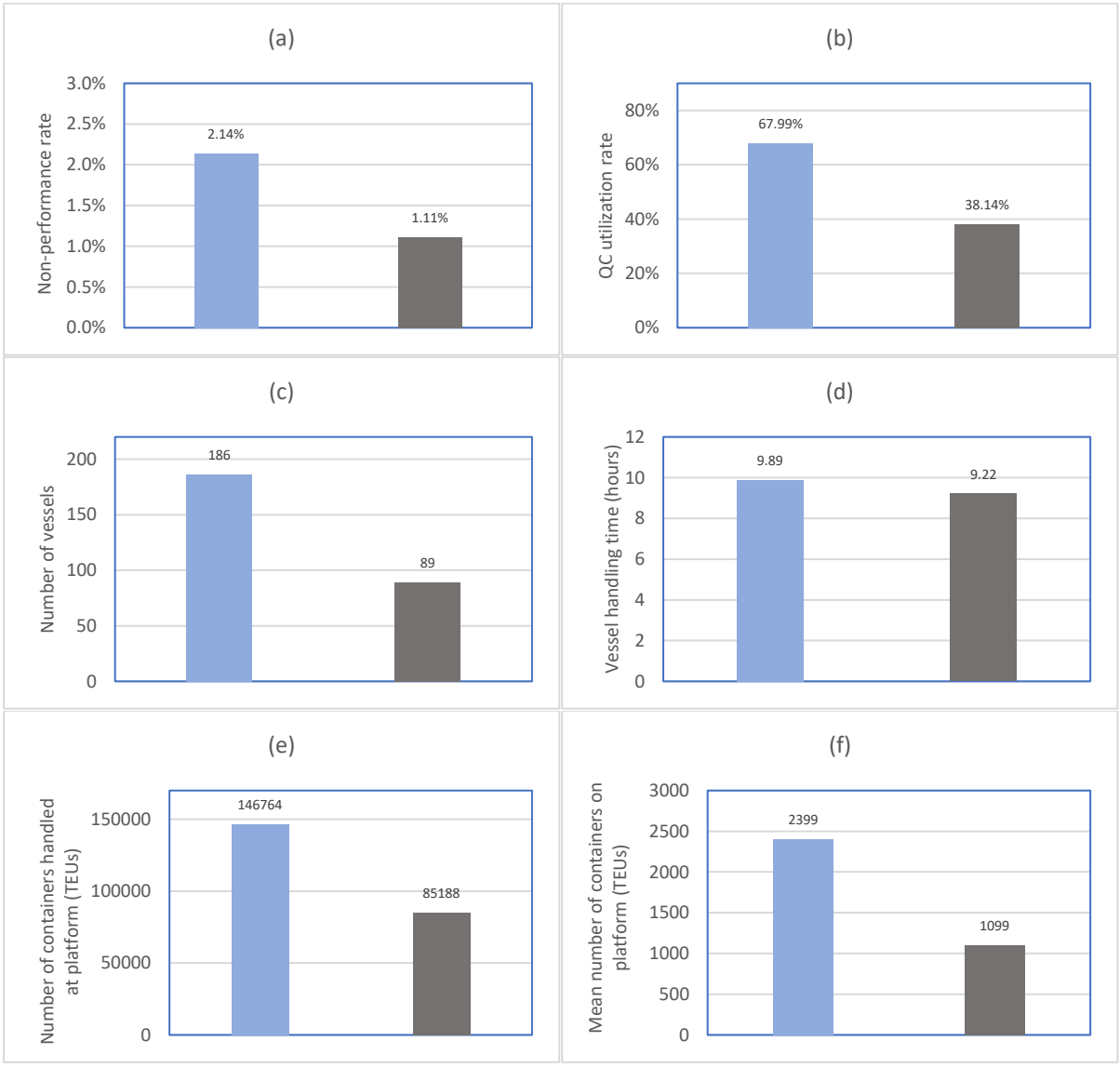
In this scenario, the growth factor is accommodated in the interarrival time of the vessels. The new interarrival time can be calculated as follow:

$$interarrivaltime_{2030} = \frac{interarrivaltime_{2017}}{growth\ factor_{2017-2030}} = \frac{100\ minutes}{2.4} = 41.67 \tag{18}$$

It is also assumed that the distribution of the interarrival time becomes wider. This assumption is accommodated by increasing the standard deviation. So, the interarrival time in this scenario is set to a non-negative normal distribution $N(\mu, \sigma^2)$ with mean $\mu = 40\ minutes$ and standard deviation $\sigma^2 = 40\ minutes$. With the new interarrival time, there are 1150 vessels and 1.7 million TEUs generated during the 6-week period. The simulation results are given in Table 5.6, while the comparison to those of the reference configuration ($\mu = 100\ minutes$, $\sigma^2 = 20\ minutes$) is shown in Figure 5.7.

Table 5.6 Results of different interarrival time

Platform performance	**Interarrival time ($N(\mu, \sigma^2)$):**	
	$\mu = 40$ *minutes*, $\sigma^2 = 40$ *minutes*	**$\mu = 100$ *minutes*, $\sigma^2 = 20$ *minutes* (ref.)**
Non-performance:	2.14% *(+1.03%)*	1.11%
Mean QC utilization rate:	67.99% *(+29.85%)*	38.14%
Vessel handled:	186 *(+1.63%)*	89
Mean vessel handling time (h):	9.89 *(+7.27%)*	9.22
Containers handled (TEUs):	146764 *(-0.4%)*	85188
Mean # of TEUs in stack (TEUs):	2399 *(+118%)*	1099
Max. # of TEUs in stack (TEUs):	6202 *(+14.36%)*	5423
Mean container dwell time (days):	0.681 *(+28.25%)*	0.531



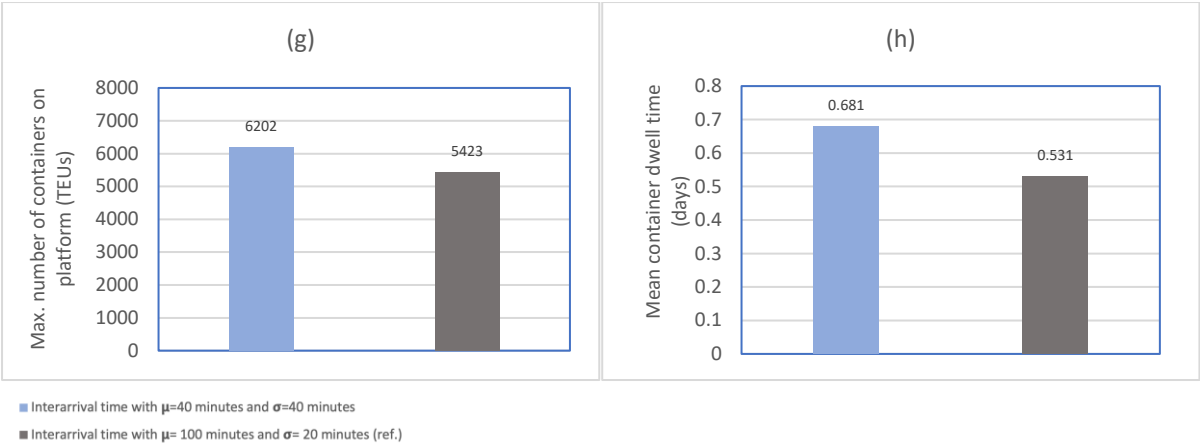


Figure 5.7 Comparison of the platform's performance with different vessel interarrival time

In general, the new of interarrival time doubles the S@S platform's share to the non-performance rate and escalates the QC utilization rate by about 30%. The statistics of the platform's stack also increase. However, the percentages of vessels and containers that are handled at the platform do not change significantly.

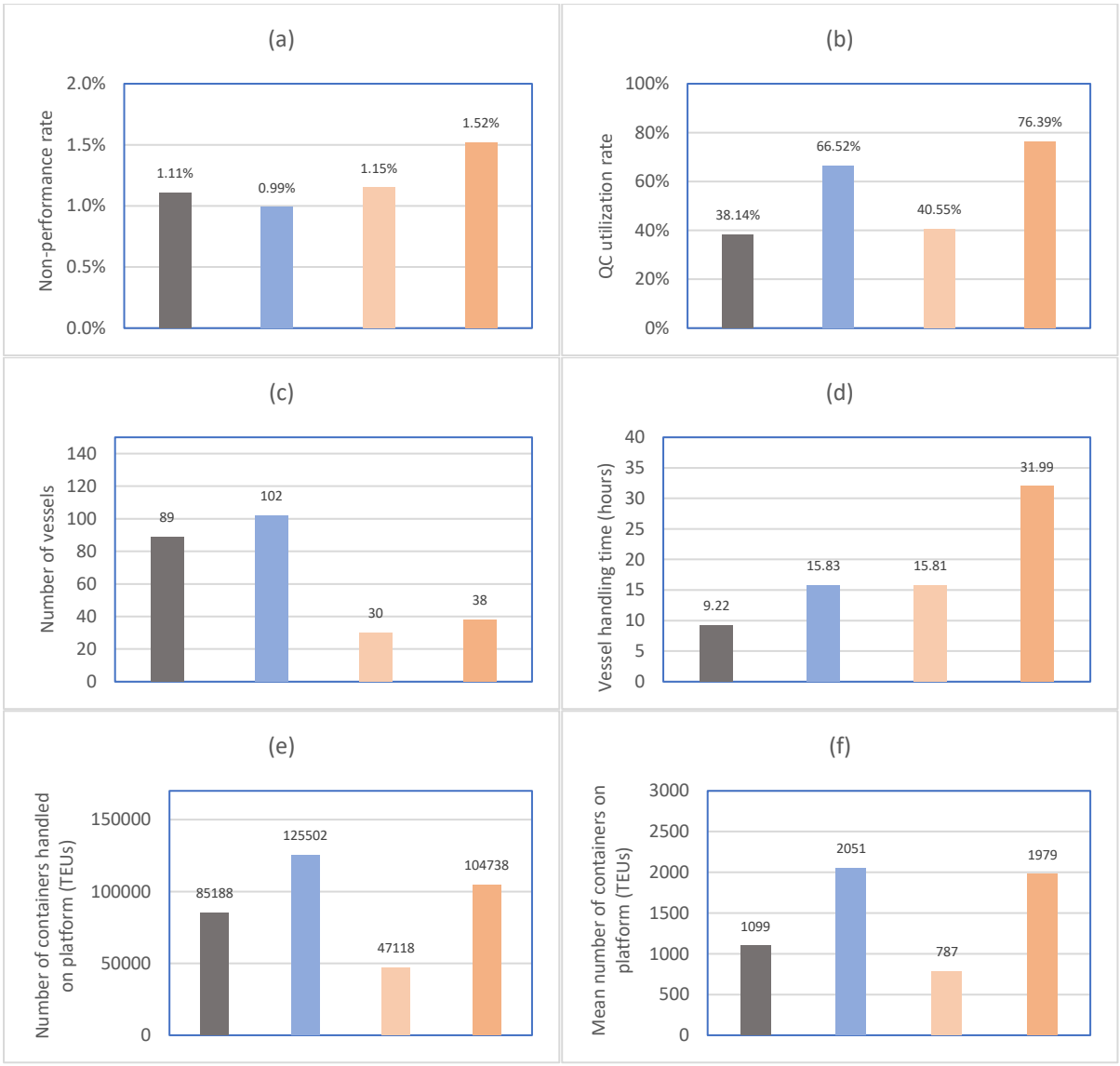
5.6.2 Increase of vessels' call sizes

In this scenario, the growth factor is assumed to affect the call size of the vessels. This is done by multiplying each of the bin classes in the call size distributions (Table 4.4) by the growth factor. If the distribution of the vessels' capacity (Table 4.2) is not changed along with the call size distribution, the increase of the call sizes will be bounded by the vessel capacity. In this way, the number of containers that are generated within the simulation period would not reach the predicted value. Therefore, to accommodate the increase of the call sizes, the vessel capacity distribution is also shifted by multiplying the mean of this distribution by the same growth factor. However, the upper bound for the capacity distribution is set to stay the same (21,000 TEUs). In addition, two other simulations are conducted with the same increase in the call sizes, but different vessel size preferences. In these extra simulations, the S@S platform is set to only handle ULCS. With these modifications, there are 604 vessels and 1,823,901 TEUs that are generated within the simulation period. The simulation results are shown in Table 5.7, while the comparison of these scenarios to the reference case is illustrated in Figure 5.8. The increase of the vessel's call sizes causes a rise in both the number of vessels and containers that are handled at the platform. However, the percentages decrease by about 2%. With this demand scenario, the mean QC utilization rate and the mean vessel handling time at the platform also increase significantly. These are the effects of having the same number of vessels with doubled call sizes. It is also observed that the S@S platform's share to the non-performance rate slightly decreases.

Table 5.7 Results of different call size scenarios

Platform performance	**Call size scenarios & vessel allocation strategies**			
	2017 demand, ≤ 6,000 TEUs only (ref.)	**2030 demand, ≤ 6,000 TEUs only**	**2017 demand, ULCS only**	**2030 demand, ULCS only**
Non-performance:	1.11%	0.99% *(-0.12%)*	1.15% *(+0.04%)*	1.52% *(0.41%)*
Mean QC utilization rate:	38.14%	66.52% *(+28.38%)*	40.55% *(+2.41%)*	76.39% *(+38.25%)*
Vessel handled:	89	102 *(+2.12%)*	30 *(-9.64%)*	38 *(-8.33%)*
Mean vessel handling time (h):	9.22	15.83 *(+71.69%)*	15.81 *(+71.47%)*	31.99 *(+246%)*
Containers handled (TEUs):	85188	125502 *(-2.15%)*	47118 *(-4.04%)*	104738 *(-3.28%)*
Mean # of TEUs in stack (TEUs):	1099	2051 *(+86.62%)*	787 *(-28.39%)*	1979 *(+80.07%)*
Max. # of TEUs in stack (TEUs):	5423	6916 *(+27.53%)*	4853 *(-10.51%)*	6952 *(+28.19%)*
Mean container dwell time (days):	0.531	0.681 *(+28.24%)*	0.667 *(+25.61%)*	0.772 *(+45.39%)*

At the same time, in both 2017 and 2030 demand scenarios, when the S@S platform only handles ULCS, the percentages of the vessels and containers that are handled at the platform decrease. This is due to two reasons: a) there are only about 5% of ULCS from the total number of vessels, and b) the ULCS do not necessarily have large call sizes. Moreover, compared to when the S@S platform handles vessels ≤ 6,000 TEUs, the S@S platform's share to the non-performance rate is always higher. This could mean that most of the export containers that are associated with the vessels which are handled at the platform are not loaded to the vessels on time.



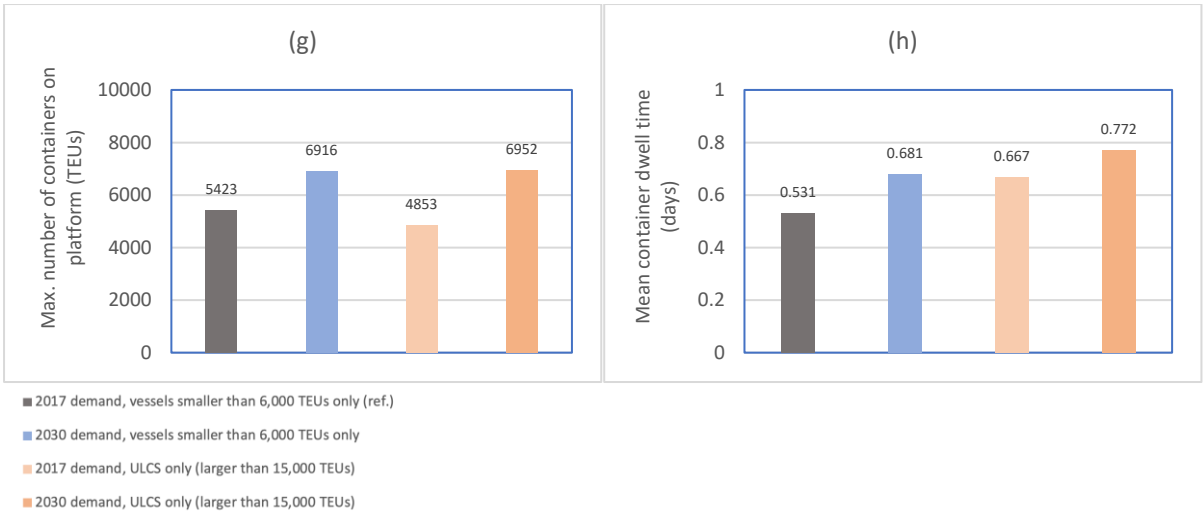


Figure 5.8 Comparison of the platform's performance with different call size scenarios and vessel allocation strategies

5.7 Comparing different maximum storage capacity of the platform

In this research, it assumed that the allocation of the vessels that visit the port-platform system takes into account the maximum storage capacity of the S@S platform. In this way, when the platform's capacity is fully occupied (or booked), vessels would not be allocated to the platform. A capacity of 10,000 TEUs is used as the reference value for the platform's maximum storage capacity. Some simulations are conducted with both 2017 and 2030 demand scenarios to see the effect of the different value of maximum storage capacity to the performance of the platform.

Table 5.8 shows the results of three simulations with 2017 demand scenario and different platform's maximum storage capacity. Though not significant, fewer vessels and containers are handled at the platform when the platform's maximum storage capacity is set to 5,000 TEUs. This configuration also increases the mean QC utilization rate and mean vessel handling time at the platform. However, the dwell time of the containers decreases. On the other hand, the platform's performance does not change when the maximum storage capacity is set to 20,000 TEUs. This indicates that with the 2017 demand scenario, the allocation of the vessels are not constrained by the platform's storage capacity when it is set to 10,000 TEUs. Figure 5.9 shows the comparison of the different platform storage capacity for the 2017 demand scenario.

Table 5.8 Results of different maximum storage capacity for the 2017 demand scenario

Platform performance	**Demand scenario and maximum capacity of the platform**		
	2017 demand, 5,000 TEUs	**2017 demand, 10,000 TEUs**	**2017 demand, 20,000 TEUs**
Non-performance:	1.13% *(+0.02%)*	1.11%	1.11% *(0%)*
Mean QC utilization rate:	41.64% *(+3.5%)*	38.14%	38.14% *(0%)*
Vessel handled:	87 *(-0.33%)*	89	89 *(0%)*
Mean vessel handling time (h):	9.8 *(+6.29%)*	9.22	9.22 *(0%)*
Containers handled (TEUs):	84583 (-0.06%)	85188	85188 *(0%)*
Mean # of TEUs in stack (TEUs):	1029 *(-6.37%)*	1099	1099 *(0%)*
Max. # of TEUs in stack (TEUs):	5423 *(0%)*	5423	5423 *(0%)*
Mean container dwell time (days):	0.497 *(-6.4%)*	0.531	0.531 *(0%)*

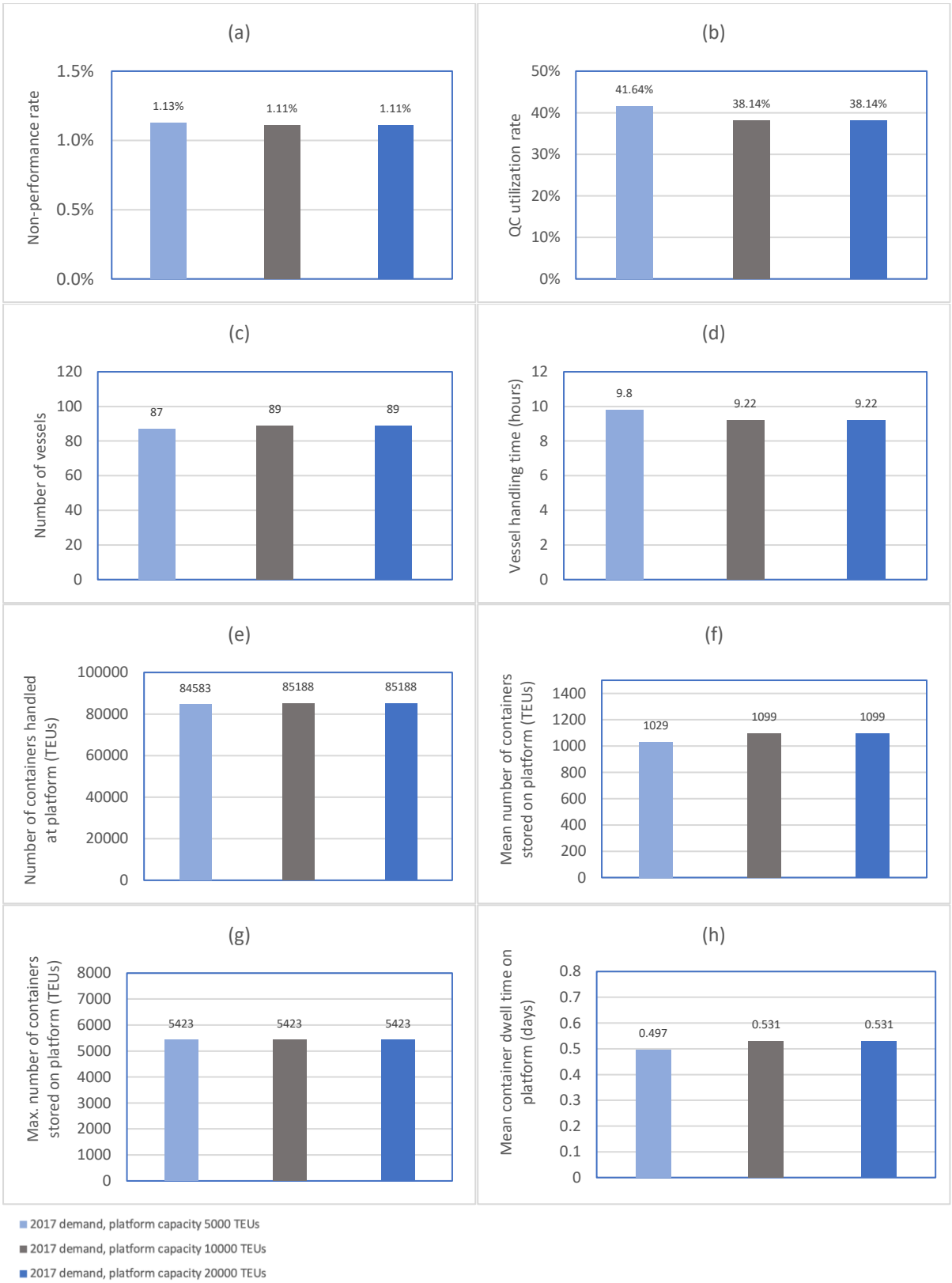


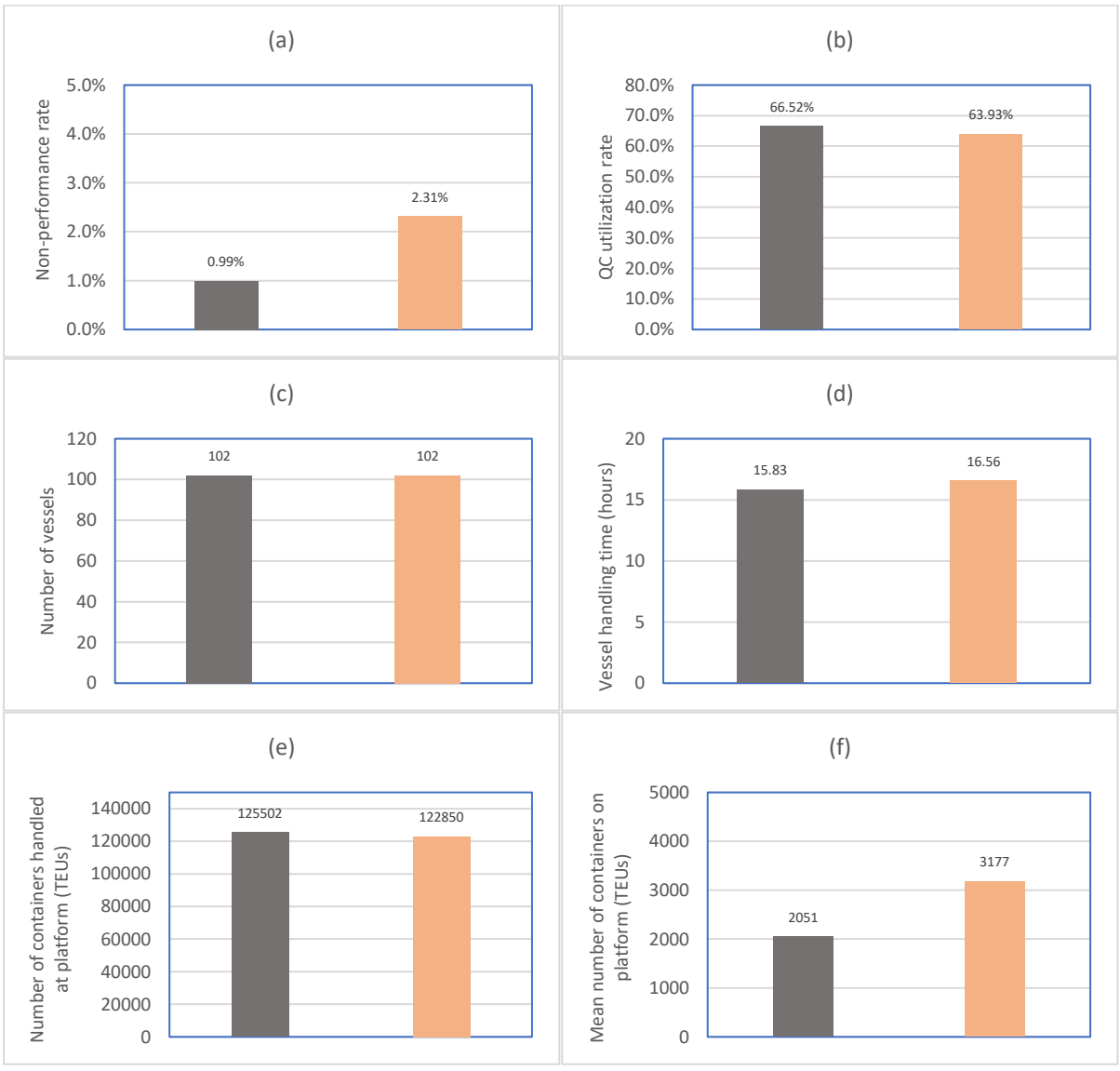
Figure 5.9 Comparison of the platform's performance with 2017 demand scenario and different maximum platform capacity

In contrast, Table 5.9 shows that with the 2030 demand scenarios, a higher value of the platform's maximum storage capacity shifts the performance of the platform. When the platform's capacity is set to 10,000 TEUs, some vessels that have large call sizes choose to not to call to the platform because the platform's capacity would not suffice their call sizes. It can be seen that the platform handles 0.15% less containers when the storage capacity is

doubled to 20,000 TEUs. This indicates that the export containers that are associated with the vessels which are handled at the port do not reach the platform on time. Therefore, the platform's share to the non-performance rate also increase. These phenomena are illustrated in Figure 5.10.

Table 5.9 Results of different maximum storage capacity for the 2030 demand scenario

Platform performance	Demand scenarios and maximum capacity of the platform	
	2030 demand, 10,000 TEUs	2030 demand, 20,000 TEUs
Non-performance:	0.99%	2.31% *(+1.32%)*
Mean QC utilization rate:	66.52%	63.93% *(-2.59%)*
Vessel handled:	102	102 *(0%)*
Mean vessel handling time (h):	15.83	16.56 *(+4.61%)*
Containers handled (TEUs):	125502	122850 *(-0.15%)*
Mean # of TEUs in stack (TEUs):	2051	3177 *(+35.44%)*
Max. # of TEUs in stack (TEUs):	6916	5767 *(-19.92%)*
Mean container dwell time (days):	0.681	1.07 *(+57.12%)*



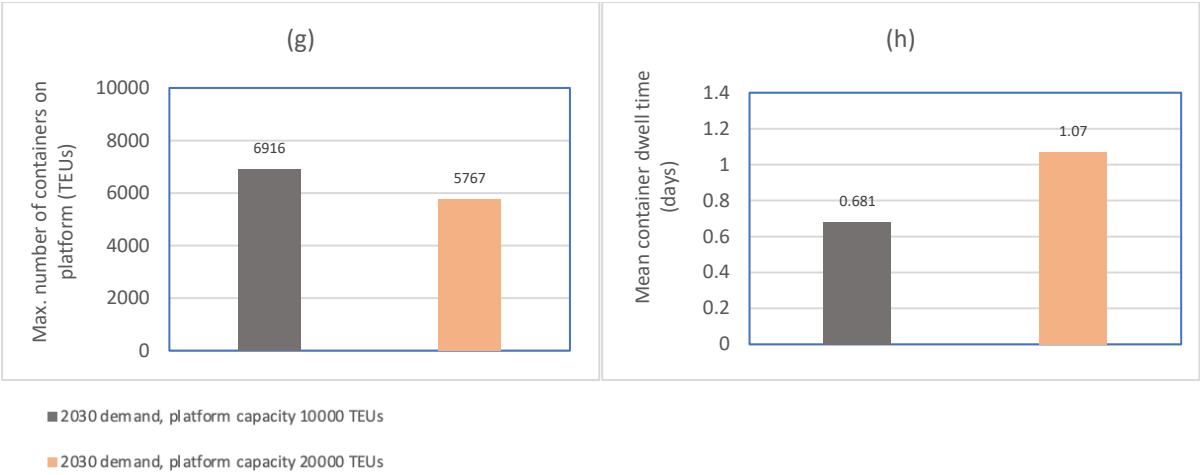


Figure 5.10 Comparison of the platform's performance with the 2030 demand scenario and different maximum platform capacity

Though the percentage of vessels that are handled at the platform does not change, it is indicated from the increase of the vessel handling time that the call sizes of the vessels that visit the platform increase when the storage capacity is set to 20,000 TEUs.

5.8 Comparing different sea-states scenarios

To have insights on how the sea states affect the handling capacity of the S@S platform, three simulations are conducted with different sea states scenarios. The scenario that has been constructed (see Section 5.9) is used as the reference scenario. The other two scenarios are extreme scenarios. One scenario assumes that the sea states are always in bad conditions. This is done by setting the *wind_factor* and *seawave_factor* to 0.3 and 0.2, respectively. Meanwhile, the other scenario assumes that the sea states are always fine.

The results of the simulation with these three sea states scenarios are shown in Table 5.10. Unfortunately, there is no available information on how an offshore terminal would perform with respect to the scenarios. Thus, it is impossible to validate the result of these simulations. Comparison of the platform's performance with respect to these scenarios are presented in Figure 5.11.

Table 5.10 Results of different sea states scenarios

Platform performance	**Sea states scenarios**		
	Bad sea-states	**Real data-based sea states (ref.)**	**Fine sea-states**
Non-performance:	8.49% *(+7.38%)*	1.11%	0.77% *(-0.34%)*
Mean QC utilization rate:	44.93% *(+6.79%)*	38.14%	40.73% *(+2.59%)*
Vessel handled:	92 *(+0.49%)*	89	84 *(-0.82%)*
Mean vessel handling time (h):	10.57 *(+14.64%)*	9.22	9.94 *(+7.81%)*
Containers handled (TEUs):	64953 *(-2.14%)*	85188	90558 *(+0.57%)*
Mean # of TEUs in stack (TEUs):	2295 *(+108%)*	1099	880 *(-19.93%)*
Max. # of TEUs in stack (TEUs):	7626 *(+40.62%)*	5423	4321 *(-20.32%)*
Mean container dwell time (days):	1.468 *(+176%)*	0.531	0.400 *(-24.67%)*

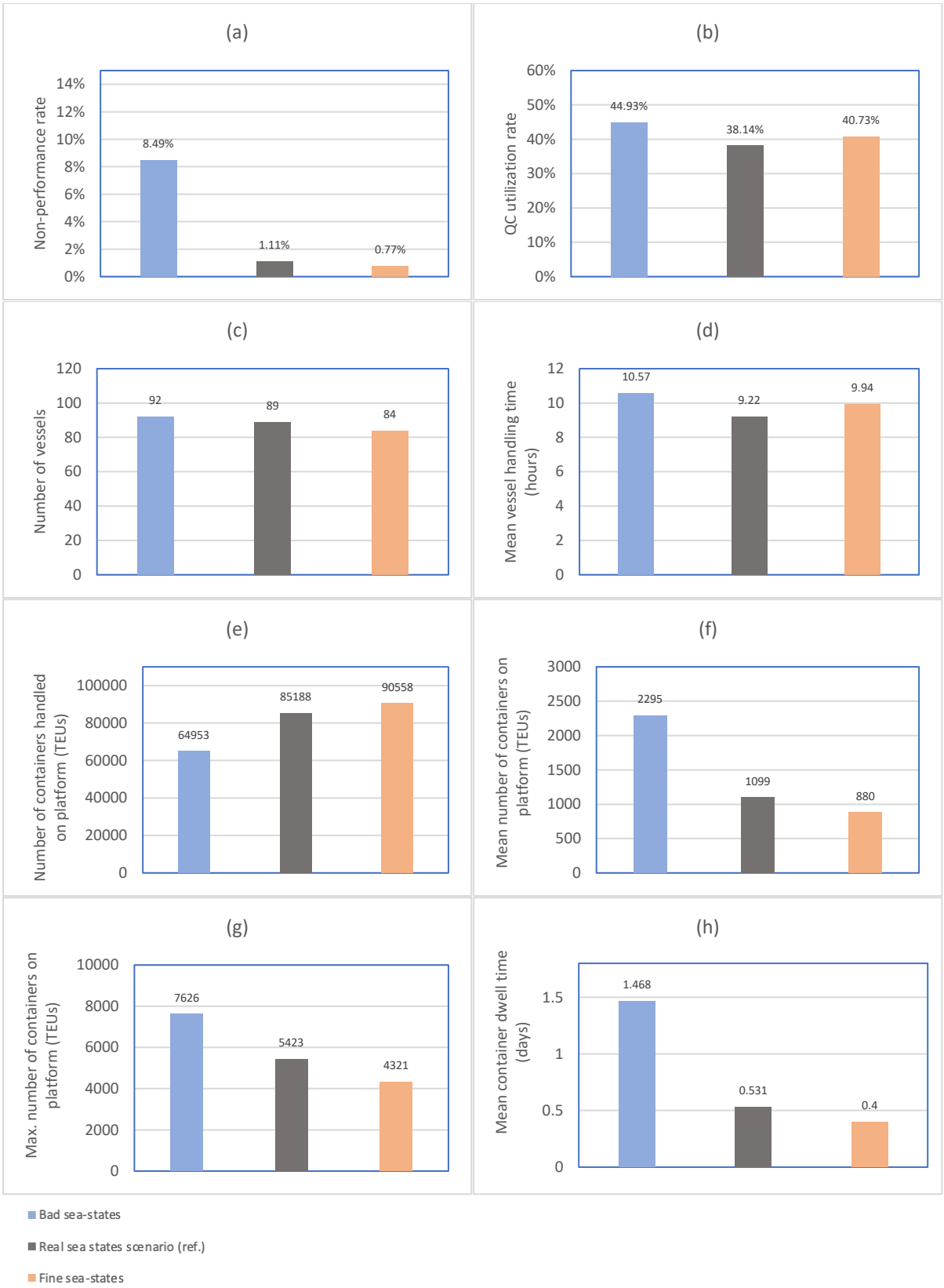


Figure 5.11 Comparison of the platform's performance with respect to different sea states scenarios

Nevertheless, the table and the figures show that the S@S platform performs best in the fine sea states scenario in terms of the number of containers that are handled and stored at the platform. In this scenario, the S@S platform handles 9.6% of the total generated containers. This percentage is 0.57% higher than the percentage in the simulation with the real sea-state scenario. Due to the constantly fine sea states, the containers spend less than half a day in

average on the platform, and the maximum number of containers that are stored on the platform does not reach 5,000 TEUs. The S@S platform's share to the non-performance rate is also lowest in this scenario. Moreover, the sea states directly affect the crane's and barge shuttle's operation. In this way, these states influenced the crane occupancy and the number of containers that are currently stored on the S@S platform, which both are used in the vessel allocation algorithm. This explains why in the fine scenario the S@S platform handles fewer vessels despite the increased container percentage.

5.9 Summary of the chapter

This chapter has evaluated the performance of some scenarios of coordination between the port and the S@S platform. In Section 5.1, an experimental plan is constructed in order to evaluate the different configurations and scenarios of the port-platform system. In Section 5.2, the result of one of the case simulation is compared to historical data and other research in order to validate the model. In Section 5.3, different barge route configurations and vessel size preferences are evaluated. Finally, some sensitivity analyses are conducted using the model to see how the performance is affected by some other configurations and scenarios, e.g. different number of QC on the platform, different location of the platform, different demand scenarios, different storage capacity of the platform, and different sea states scenarios. All of these are presented in Section 5.3 to 5.8.

Chapter 6 Conclusion

This last chapter concludes the research. Section 6.1 summarizes all the answers to the sub-questions as a framework of the answer to the main research question, and Section 6.2 gives recommendations for research in the future.

6.1 Conclusion

"What are the components of the existing port system? How are they configured, and how do they currently perform? Who are the stakeholders, and what kind of services do they provide for handling incoming containers?"

The Port of Antwerp is investigated as the object of interest for this research in Chapter 2. The Port of Antwerp has 24 terminals, but only five of them are specifically designed for handling sea-going container vessels. These terminals have different operators and different number of QCs. While all of them are connected to the hinterland by road, rail and, waterway network, the intra-port container distribution is also done via these three networks. The Instream initiatives are the ones promoting inland shipping programs as the port is trying to shift its modal split more to the inland waterway shipping. These initiatives have introduced a central coordination point to distribute the call of the vessels, and a barge shuttle service to provide a connection between the terminals.

"What are the processes involved during the handling of containers within a port? What are the performance indicators? What are the commonly used methods for the performance evaluation of a port? How would the performance of an offshore container terminal affected by the sea-states?"

An overview of the general container terminal processes is illustrated in Chapter 3. In general, containers arrive at a terminal by the sea-going vessels and the hinterland connections. Sea-going vessels are loaded/unloaded by the QCs. Containers are stored as stacks in the terminal while waiting to be either a) distributed to another terminal by an ITT system, b) loaded to a sea-going vessel, or c) delivered to the hinterland by one of the transport modes.

A port has a lot of parameters and performance indicators. In order to show the S@S platform affects the handling capacity of the Port of Antwerp, this research use these indicators: a) QC utilization rate, b) number of vessels and containers handled at the terminals, c) vessel handling time at the terminals, c) container dwell time in the terminals' stacks, d) mean and maximum number of containers in the terminals' stack, and d) the non-performance rate. The non-performance indicator indicates when a container is delivered later than its due time.

Different methods have been used to evaluate the performance of a port. There are two kinds of method in general: analytic methods and simulation methods. This research chooses to use discrete event simulation approach, for these following reasons: a) it can be used to

evaluate different alternatives, b) it can generate stochastic behaviour, and c) it can be visualized by animation to assist the verification and validation phase of the model.

Offshore container handling operations have been studied from many years ago. It is found from the literature that the dynamics of offshore container handling is influenced by the sea wave motions. The offshore platform and the vessel that is currently being handled respond to the sea waves in different motions. There has been a lot of research conducted to analyze the responses of both the offshore platform and the vessels during the handling operation. There has also been some research which compare the performance of a land-based terminal to an offshore terminal. Most studies stated that while it is true that the sea wave motions generate motion responses, the effect of these responses is still within the operational limits. One research claims that the offshore container terminal can operate with significant sea wave height up to 3 meters. Meanwhile, research on how an offshore container terminal would affect the handling capacity of an existing port is still scarce.

"How should the port-platform system be modelled, and how should the platform coordinate with the port in order to have an integrated container handling service? What are the configuration and scenarios to be evaluated using the model? Does the model behave the way the user intended it to be?"

The description of how the simulation model is made is presented in Chapter 4. The model takes into account six terminals of the port and one offshore platform. The number of QCs in each terminal varies, and the number of QCs that can be assigned to a vessel depends on the vessel's size. The container distribution is assumed to be done solely by the barge shuttle. The input of the model are: a) terminal configuration, b) container handling demand scenarios, c) barge route configuration, and d) weather scenarios. The output is the performance indicators which have been stated in the previous paragraph. The objects in the model are: VesselGenerator, Containers, Vessels, Terminals, Barges, QuayCranes, BargeCranes, and Weather.

An algorithm is proposed to allocate the Vessels to one of the Terminals. The algorithm considers three parameters regarding the handling operation at the platform: a) the size of the vessels to be handled at the platform, b) the storage capacity of the platform, and c) QC utilization rate of the terminals. The platform is assumed to be installed as an extension to the existing port. Therefore, vessels will prefer to be handled at the port if all the criteria are fulfilled.

Two barge route strategies are constructed. In the first route strategy, the barge shuttle has a loop route that goes to each of the terminals once in a day. The second strategy uses two different routes: one loop route around the port area, and another route which connects the port and the platform. The container handling demand scenario is constructed based on a historical dataset of vessels visiting the Port of Antwerp in the year 2017. This dataset has been analyzed, so sizes of these vessels and their call sizes distributions are known. Meanwhile, the sea states scenario is constructed based on a historical dataset of the wind speed and significant sea wave height at the location of the platform.

The model has been verified by conducting several test runs and checks. The trace monitor and animation window are used in order to observe the behaviour of the model during the test runs and checks.

"Does the model represent the behaviour of the actual system? If so, how does the platform affect the performance of the port when different configurations and scenarios are implemented?"

The simulation model has been validated in Chapter 5 by comparing the result of one simulation without the platform installed to several historical data and similar studies. It is found that the mean QC utilization rate of the model does not reach the expected value. Though, one report shows that the utilization rate of the Port of Antwerp is relatively low when compared to those of other ports, and the value mentioned in the report matches with

the results of the model. At the same time, when the QuayCrane's and BargeCrane's *move_time* are set to realistic values, the non-performance rate of the model becomes very high. A dedicated planning and control scheme is needed to allocate the vessels based on their departure time, and to sort containers based on their due times. As this scheme is out of the scope, this research tries to compensate the need of this scheme by lowering the QuayCrane's and BargeCrane's *move_time* values. Though the *move_time* values are too optimistic, the results of the simulation model have been validated by comparing the statistics of the terminal stack to the values stated in the S@S memorandum document. It is proven from this memorandum document that the simulation results can still be used to see the trends on how the performance of the port is affected by the presence of the platform.

Five cases are constructed from the combination of two kinds of barge route strategy and two kinds of vessel allocation strategy. The simulation results of these five cases are used to determine the preferred barge route strategy between the port and the platform, as well as the vessel allocation strategy at the platform. Furthermore, the preferred strategies are used to conduct sensitivity analysis on several other configurations and scenarios, e.g., a) different QuayCrane's and BargeCrane's *move_time* values, b) different number of QCs at the platform, c) different platform locations, d) different demand scenarios, e) different maximum storage capacity of the platform, and f) different sea states scenarios. The results of the sensitivity analysis are used to answer the main question of this research:

"How would the modular offshore platform, when implemented as a container transhipment/storage hub, affects the handling capacity of an existing port?"

In general, the installation of the platform will increase the total handling capacity of the port. Some vessels and containers will be handled at the platform, and this will reduce the mean QC utilization rate and the need for storage capacity at the port terminals. Moreover, when the platform is installed, some sea-going vessels would not have to sail to the port to have their containers handled. In this way, the vessels that are handled at the platform could save about 8 hours of sailing time to/from the port. This might also reduce the vessel traffic at the Scheldt river. However, as it is assumed that there is no direct transhipment to the hinterland terminals, all the containers that are handled at the platform should be transferred to the port before being delivered to their hinterland destinations. As a consequence, the platform installation will increase the non-performance rate of the system.

Based on the simulation results, this research suggests to implement the two-barge-route strategy as the container distribution scheme between the port and the platform and recommends the platform to only handle vessels with capacity ≤ 6,000 TEUs. These two strategies are preferred for the reasons that they provide a better platform performance, high percentages of vessels (14.54%) and containers (9.03%) that are handled at the platform, as well as a relatively low non-performance rate (1.11%) when compared to the other cases.

It is observed from the simulation results that adding five more QCs on the platform would reduce the non-performance rate of the platform by 0.11% and the mean QC utilization rate on the platform by 3.63%. Meanwhile, the percentages of vessels and containers that are handled at the platform would increase by 2.5% and 2%, respectively. Despite, it is important to note that in order to have more QCs, the platform would need more space, and therefore more modules would be needed for the platform.

In this research, three different locations of the platform are compared by changing the sailing time between the port and the platform. A closer location of the platform would reduce the non-performance rate of the platform, the number of containers that are stored on the platform, and the mean container dwell time on the platform. In contrast, a further location of the platform would increase these performance indicators. Though the location of the platform is not taken into account in the vessel allocation algorithm, it directly affects the container distribution between the port and the platform. Thus, the location of the platform indirectly affects the QC occupancy of the terminals. As the QC utilization rate of the terminals is included in the vessel allocation algorithm, the number of vessels that are handled at the platform will also change when the location of the platform is changed.

In the future, the container handling demand at the Port of Antwerp will increase. It is intended that the installation of the platform should accommodate this future container handling demand. Two kinds of future container handling demand scenarios are examined: a) increase of the vessel's interarrival time, and b) increase of the vessel's call sizes. These future demand scenarios are constructed based on a future economic projection for the year 2030 that is included in the ITT project for Maasvlakte 1&2 in Port of Rotterdam.

When the vessel's interarrival time is doubled, the numbers of vessels and containers that are handled at the platform increase. Accordingly, the mean QC utilization rate of the platform also increases by about 30%. However, the ratios of the number of vessels and containers that are handled on the platform to the total number of vessels and containers that are generated during the simulation period do not change significantly. Moreover, the doubled interarrival time also increase the non-performance rate by 1.03%, the mean vessel handling time at the platform by 7.27%, as well as the mean container dwell time on the platform by 28.25%. Similar phenomena are observed when the vessel's call sizes are increased. The numbers of vessels and containers that are handled at the platform become higher, resulting in an increase of the mean QC utilization rate on the platform by 28.38%, and an increase of the mean container dwell time on the platform by 28.24%. However, when the demand growth is accommodated by increasing the vessel's call sizes, the mean vessel handling time on the platform grows significantly by 71.69%. The increase of these performance indicators are natural, because the container distribution service rate remains constant while there are more vessels and containers in the system.

Furthermore, it is found from the results that in the 2017 demand scenario, the platform's performance would not be affected if the maximum storage capacity of the platform is increased to 20,000 TEUs. In contrast, when the maximum storage capacity of the platform is set to 20,000 TEUs in the 2030 demand scenario, vessels that have larger call sizes can visit the platform and have their containers stored on the platform. This is indicated by the constant number of vessels that are handled at the platform and the increase in the number of containers that are handled at the platform.

Lastly, three different sea states scenarios are evaluated with the simulation model. Apart from the real data-based sea states scenario, two other extreme sea states scenarios are constructed: a) bad sea states scenario, and b) fine sea states scenario. It is shown from the simulation results that when compared to the constantly fine sea states scenario, the real data-based sea states scenario only reduces the number of containers that are handled at the platform by 0.57%. Though there is no available historical data that can be used to validate the results, this value corresponds with previous studies which state that the sea states will only reduce the handling capacity of an offshore container terminal by no more than 1%.

6.2 Future work recommendations

This research has developed a model to evaluate the performance of a port when the container distribution service is extended by the presence of an offshore modular platform. This tool can be used by the S@S project partners to evaluate other configurations and scenarios that have not been described in this research. Though this research focuses on the Port of Antwerp as the case study, this model can be used for the evaluation of other port by modifying the terminal and barge route configurations.

The model development phase is constrained by the time limitation of this research. Therefore, the level of details that are given in this model is also limited. In order to have a more valid model, it is recommended for future research to use the model and expand the program code by introducing other important parameters for the vessel allocation algorithm. For instance, instead of only the available QCs, the vessel can also decide where to call by considering the quay length of the terminal, or by taking into account the weather forecast at the location of the platform. Moreover, while there is a random characteristic in the generated demand, this research only used the FIFO algorithm to handle the incoming vessels

and containers. There is a huge probability that the vessels are not fully handled, and the containers are delivered later than their due times. The model will generate more valid outcomes if better planning and control are introduced to allocate the vessels based on their departure times and to sort the containers based on their due times. It is also suggested to integrate this model with a traffic modelling of the Scheldt river, so the operations of the barge shuttle can be evaluated more accurately.

In reality, a direct transhipment scheme is more preferred for the offshore modular platform, as barges from the hinterland can directly visit the platform instead of having to go to the port. In this way, the installation of the platform will have a more significant impact on the overall logistics operations. Therefore, this research suggests future work to investigate the extension of the coordination with the hinterland terminals. Furthermore, a cost-benefit analysis can be conducted to have insights on the economic aspects of the platform installation.

Bibliography

[1] "Multi-use affordable standardised floating Space@Sea | Projects | H2020 | CORDIS | European Commission." [Online]. Available: https://cordis.europa.eu/project/rcn/212413_en.html. [Accessed: 26-Jun-2018].

[2] "About SPACE@SEA - SPACE@SEA." [Online]. Available: https://spaceatsea-project.eu/about-space-at-sea. [Accessed: 26-Jun-2018].

[3] "Delft University of Technology - SPACE@SEA." [Online]. Available: https://spaceatsea-project.eu/partners/delft-university-of-technology. [Accessed: 26-Jun-2018].

[4] Port of Antwerp, "Yearbook of Statistics 2017," Antwerp, 2017.

[5] "Containers | Port of Antwerp." [Online]. Available: https://www.portofantwerp.com/en/containers. [Accessed: 05-Jul-2018].

[6] "Deepening of the Scheldt: new regulations for upstream and downstream navigation | Port of Antwerp." [Online]. Available: https://www.portofantwerp.com/en/deepening-scheldt-new-regulations-upstream-and-downstream-navigation. [Accessed: 05-Jul-2018].

[7] "How much bigger can container ships get? - BBC News." [Online]. Available: https://www.bbc.com/news/magazine-21432226. [Accessed: 27-Jan-2019].

[8] Antwerp Port Authority, "Instream: Smart and efficient inland navigation," Antwerp, 2015.

[9] "Find your container terminal in the port | Port of Antwerp." [Online]. Available: https://www.portofantwerp.com/en/intraport-terminal-tool. [Accessed: 05-Jul-2018].

[10] "Optimising of container barge handling | Port of Antwerp." [Online]. Available: https://www.portofantwerp.com/en/optimising-container-barge. [Accessed: 25-Nov-2018].

[11] PSA Antwerp, "PSA Noordzee Terminal Factsheet." [Online]. Available: https://www.psa-antwerp.be/en/file/14016/download?token=CjGYQ1-A. [Accessed: 09-Aug-2017].

[12] PSA Antwerp, "PSA Europa Terminal." [Online]. Available: https://www.psa-antwerp.be/nl/file/5061/download?token=4wP81_Zd. [Accessed: 09-Aug-2017].

[13] PSA Antwerp, "MSC PSA European Terminals (MPET)." [Online]. Available: https://www.psa-antwerp.be/en/file/13977/download?token=ycxXPe_0. [Accessed: 09-Aug-2017].

[14] "Antwerp Gateway | DP World Antwerp." [Online]. Available: http://www.dpworldantwerp.com/our-businesses/antwerp-gateway. [Accessed: 09-Aug-2018].

[15] "Working to maximise efficiency of goods transport | Port of Antwerp." [Online]. Available: https://www.portofantwerp.com/en/working-maximise-efficiency-goods-transport. [Accessed: 06-Jul-2018].

[16] "Antwerp Port Shuttle - Container transport per binnenschip." [Online]. Available: http://www.apsantwerp.be/nl/static_pages/premium_barge_service. [Accessed: 23-Nov-2018].

[17] I. F. A. Vis and R. De Koster, "Transshipment of containers at a container terminal: An

overview," *Eur. J. Oper. Res.*, vol. 147, no. 1, pp. 1–16, 2003.
[18] J. A. Ottjes, M. B. Duinkerken, J. J. M. Evers, and R. Dekker, "Robotised Inter Terminal Transport of containers: A simulation study at the Rotterdam Port Area," in *Proceedings of the 8th European Simulation Symposium*, 1996.
[19] "Inter Terminal Transport - Scientic Area of Rudy R. Negenborn." [Online]. Available: http://www.negenborn.net/rudy/projects_itt.html. [Accessed: 13-Jul-2018].
[20] R. R. Negenborn and M. B. Duinkerken, "Definition of common parameter values required for ITT system design," Delft, 2014.
[21] E. J. Gerritse, "Analysis for Inter Terminal Transportation demand scenarios for the Maasvlakte I and II in 2030," Delft, 2014.
[22] K. Tierney, S. Voß, and R. Stahlbock, "A mathematical model of inter-terminal transportation," *Eur. J. Oper. Res.*, vol. 235, no. 2, pp. 448–460, 2014.
[23] F. Nieuwkoop, F. Corman, R. R. Negenborn, M. B. Duinkerken, M. Van Schuylenburg, and G. Lodewijks, "Decision support for vehicle configuration determination in Inter Terminal Transport system design," *Proc. 11th IEEE Int. Conf. Networking, Sens. Control. ICNSC 2014*, pp. 613–618, 2014.
[24] H. J. L. Schroer, F. Corman, M. B. Duinkerken, R. R. Negenborn, and G. Lodewijks, "Evaluation of inter terminal transport configurations at Rotterdam Maasvlakte using discrete event simulation," in *Proceedings of the 2014 Winter Simulation Conference*, 2014, pp. 1771–1782.
[25] A. Caris, C. Macharis, and G. K. Janssens, "Network analysis of container barge transport in the port of Antwerp by means of simulation," *J. Transp. Geogr.*, vol. 19, no. 1, pp. 125–133, 2011.
[26] J. A. Ottjes, H. P. M. Veeke, M. B. Duinkerken, J. C. Rijsenbrij, and G. Lodewijks, "Simulation of a multiterminal system for container handling," *Contain. Termin. Cargo Syst. Des. Oper. Manag. Logist. Control Issues*, vol. 468, pp. 15–36, 2007.
[27] M. B. Duinkerken, R. Dekker, S. T. G. L. Kurstjens, J. A. Ottjes, and N. P. Dellaert, "Comparing transportation systems for inter-terminal transport at the Maasvlakte container terminals," *Contain. Termin. Cargo Syst. Des. Oper. Manag. Logist. Control Issues*, vol. 493, pp. 37–61, 2007.
[28] A. Ballis and C. Abacoumkin, "A container terminal simulation model with animation capabilities," *J. Adv. Transp.*, vol. 30, no. 1, pp. 37–57, 1996.
[29] W.-C. Huang, T.-C. Kuo, and S.-C. Wu, "A comparison of analytical methods and simulation for container terminal planning," *J. Chinese Inst. Ind. Eng.*, vol. 24, no. 3, pp. 200–209, 2007.
[30] Y. Sukeyasu *et al.*, "New proposal of evaluation method for cargo handling efficiency on mega float container terminal facility," *Ocean. '04 Mts/IEEE Techno-Ocean '04, Vols 1- 2, Conf. Proceedings, Vols. 1-4*, pp. 1067–1072, 2004.
[31] A. Ali and H. Ligteringen, "Floating Transshipment Container Terminal," Delft University of Technology, 2005.
[32] J. Kim and J. R. Morrison, "Offshore port service concepts: Classification and economic feasibility," *Flex. Serv. Manuf. J.*, vol. 24, no. 3, pp. 214–245, 2012.
[33] A. J. Baird and D. Rother, "Technical and economic evaluation of the floating container storage and transhipment terminal (FCSTT)," *Transp. Res. Part C Emerg. Technol.*, vol. 30, pp. 178–192, 2013.
[34] M. A. Dulebenets, M. M. Golias, S. Mishra, and W. C. Heaslet, "Evaluation of the floaterm concept at marine container terminals via simulation," *Simul. Model. Pract. Theory*, vol. 54, pp. 19–35, 2015.
[35] Nederlands Normalisatie-instituut, "NEN 2018: Cranes - Loads and combinations of loads," Delft, 1983.
[36] H. Schim van der Loeff, "Quay Crane productivity at the ECT Delta terminal," Delft University of Technology, 2007.
[37] P. Chhetri, G. B. Jayatilleke, V. O. Gekara, A. Manzoni, and B. Corbitt, "Container terminal operations simulator (CTOS) – Simulating the impact of extreme weather events on port operation," *EJTIR*, no. 16, pp. 195–213, 2016.
[38] B. J. A. Pielage, J. C. Rijsenbrij, and H. Ligteringen, "Floating cranes for container handling," *2008 1st Int. Conf. Infrastruct. Syst. Serv. Build. Networks a Bright. Futur. INFRA 2008*, 2008.
[39] E. H. Kim *et al.*, "An advanced cargo handling system operating at sea," *Int. J. Control.*

Autom. Syst., vol. 12, no. 4, pp. 852–860, 2014.
[40] A. A. Shabayek and W. W. Yeung, "Effect of Seasonal Factors on Performance of Container Terminals," *J. Waterw. Port, Coastal, Ocean Eng.*, vol. 127, no. 3, pp. 135–140, 2001.
[41] M. H. Kim, B. Kumar, and J. W. Chae, "Performance Evaluation of Loading/Offloading from Floating Quay to Super Container Ship," in *Proceedings of the Sixteenth International Offshore and Polar Engineering COnference*, 2006, vol. 4, pp. 144–149.
[42] H. Y. Kang, M. H. Kim, J. H. Kim, W. S. Park, and J. W. Chae, "Offloading From Both Sides of a Super-container Ship to Land and Floating Harbor Predicted vs . Experimental Results," in *Proceedings of the Twentieth International Offshore and Polar Engineering Conference*, 2010, vol. 7, pp. 581–587.
[43] G. F. Thiers and G. K. Janssens, "A Port Simulation Model as a Permanent Decision Instrument," *Simulation*, vol. 71, no. 2, pp. 117–125, 1998.
[44] K. Braekers, A. Caris, and G. K. Janssens, "Optimal shipping routes and vessel size for intermodal barge transport with empty container repositioning," *Comput. Ind.*, vol. 64, no. 2, pp. 155–164, 2013.
[45] J. Martin, S. Martin, and S. Pettit, "Container ship size and the implications on port call workload," *Int. J. Shipp. Transp. Logist.*, vol. 7, no. 5, p. 553, 2015.
[46] Port of Antwerp, "2017 Facts and Figures," Antwerp, 2017.
[47] Ports Regulator of South Africa, "Port Benchmarking Report 2015/2016," Durban, 2016.
[48] C. Ducruet and O. Merk, "Examining container vessel turnaround times across the world," *Port Technol. Int.*, pp. 18–20, 2013.
[49] G. Mbiydzenyuy, "An Optimization Model for Sea Port Equipment Configuration," Blekinge Institute of Technology, 2007.
[50] Y. C. Yang and C. L. Lin, "Performance analysis of cargo-handling equipment from a green container terminal perspective," *Transp. Res. Part D Transp. Environ.*, vol. 23, pp. 9–11, 2013.
[51] F. F. Achterberg, "Trends in ship-to-shore container cranes," 2012.
[52] D. Steenken, S. Voß, and R. Stahlbock, "Container terminal operation and operations research - A classification and literature review," *Contain. Termin. Autom. Transp. Syst. Logist. Control Issues Quant. Decis. Support*, pp. 3–49, 2005.
[53] Z. Zenzerovic, S. Vilke, and N. Antonini, "Cost Model in Function of Optimal Capacity Planning of Port Container Terminal," pp. 1–13, 2013.
[54] B. Wiegmans, "Intermodal freight terminals: Terminal handling costs," *J. Infrastruct. Plan. Manag.*, vol. 4, no. 632, pp. 1–16, 1999.

Appendix A Research paper format of the report

Performance evaluation tool for the expansion of a port's container network by an offshore modular platform

R.A. Mulkan, D. Souravlias, M.B. Duinkerken, D.L. Schott, R.R. Negenborn
Department of Maritime and Transportation Technology
Delft University of Technology
Delft, The Netherlands

***Abstract*—The Space@Sea platform is a sustainable offshore modular platform designed for a workspace at sea as an effort in coping up with the economic growth and allocating marine resources to more efficient uses. There are four potential applications of the platform: farming, transport & logistics hub, energy hub, and living spaces. In the case of the transport & logistics application, the platform is expected to be implemented as a container transhipment/storage hub operating at sea. In this paper, a discrete event simulation model is built and used as a tool to evaluate several configurations and scenarios for the coordination between a port and the offshore modular platform in order to integrate the logistics operations on the platform to the existing port container service. The Port of Antwerp is selected for a case study.**

Keywords*—*container, terminal, port, Antwerp, performance evaluation, coordination, Space@Sea, offshore modular platform, discrete event simulation, barge routing strategy, Python, salabim

I. INTRODUCTION

The Space@Sea is a project funded by the Horizon 2020 programme of the European Commission. The project aims to develop a sustainable offshore modular platform for four kinds of application: farming, transport & logistics hub, energy hub, and living. The project was initiated in November 2017 and planned as a three-year project involving 17 project partners from all across Europe. As one of the work package in the project, the WP-9 Transport&Logistics@Sea attempts to integrate the logistics operations of the modular offshore platform to an existing port service by coordinating at the strategic, tactical, and operational levels. In order to achieve the goal, several studies are conducted to determine the essential decisions for the platform configuration, e.g., size of the platform, number of loading and unloading equipment, storage capacity, types of the vessels to be handled at the platform, as well as the coordination with the existing port system.

Currently, sea-going vessels have to transfer their containers at the Port of Antwerp that is located inland. The vessels need to make a turn into the Scheldt river at some points on their route, queue and wait for their containers to be handled at the port, then sail back to the sea to continue with their journey. The logistics operations at this port will be influenced by the presence of the platform. Instead of having to turn into the river passage, the sea-going vessels also have the choice to have their containers handled at the offshore platform and continue straight with their journey. The containers can then be picked by a dedicated connection from the port. While studies on the design and configuration of the platform have been conducted, those concerning the real-time logistics operations on and around the platform have not been initiated. There is a need to investigate how the platform would affect the performance of the existing port, and how the platform should coordinate with the port when different configurations and scenarios are implemented; for instances:

1) If the platform only handles small & medium vessels (100 – 6,000 TEUs),
2) If the platform only handles large vessels (≥ 6,000 TEUs), and
3) If the platform only handles *Ultra Large Container Vessels*.

Based on the conditions described beforehand, the main question to be answered at the end of this research is stated below:

"How would the modular offshore platform, when implemented as a container transhipment/storage hub, affects the handling capacity of an existing port system?"

This research will develop a tool to evaluate the performance of a port which container handling service is extended by the presence of an offshore modular platform. The tool can be used by the S@S partners to have insights on how the S@S platform affects the handling capacity of the Port of Antwerp when the configurations and scenarios proposed by the other S@S partners are actualized. Though this research uses the Port of Antwerp as a case study, the tool is generic enough to be used by other port authorities to compare the performance of a port when, for example, a new intra-port container distribution scheme is implemented, or when a new transhipment/storage hub is introduced within the port network. Despite the fact that there has been a lot of research regarding the operation of an offshore terminal, these research are not focused on how to coordinate the offshore terminal operation to an existing container handling service. Moreover, most of the research that looks into the operational aspects of an offshore terminal only investigated the terminal's and vessel's motion responses due to the sea wave motions. There is still a limited number of research on how these motions affect the overall handling capacity of the extended port service. Therefore, this research will try to contribute to the scientific domain by filling these literature gaps.

This research starts with the analysis of the Port of Antwerp and its components. A literature review is conducted afterwards as an effort to find theoretical backgrounds and state-of-the-art of the topics that are relevant to this research. Then, a model is developed, along with several configurations and scenarios related to the port-platform logistics operations. Finally, some experiments are conducted using the model, and the results of these experiments are analyzed to determine the effect of the platform installation to the handling capacity of the port.

II. Port of Antwerp

The Port of Antwerp is an international seaport which has acted as one of the major links of the world trade since the beginning of the 19th century. The port covers a total area of 12,068 hectares on the banks of the Scheldt River. This area is used by about 900 private companies to perform their logistics activities. Meanwhile, the port infrastructure is managed by the Antwerp Port Authority. The port handles five types of goods: containers, liquid bulk, dry bulk, breakbulk, and ro-ro. There are 24 terminals at the port area. With an overall performance rate of 40 crane movements per hour per crane, the Port of Antwerp by far has the most productive deep-sea container terminals throughout Europe. The port terminals were visited by a total of 4,289 container vessels and 59,268 hinterland barges in 2017. Within the same year, containers took about 55% from 223 million tonnes of maritime freight volume handled in the Port of Antwerp. This percentage has approximately remained the same since 2010; but the actual number of containers has increased exponentially throughout 30 years. The port is also accessible for the *Ultra Large Container Ships* (ULCS) as the biggest container vessels in the world.

A. Port infrastructure

The container facilities of the port (customs, empty depots, repair centres, etc.) are spread along with the 24 port terminals. However, sea-going container vessels can only be handled in the five maritime container terminals (Fig. 1). All the containers coming and leaving to/from Port of Antwerp must have been handled in one of these terminals. Moreover, it is foreseen that in the future these five terminals will only handle vessels with call size more than 30 containers.

With a central location and well-developed hinterland connections, the port can deliver containers to their next destinations by road transport, rail transport, or inland shipping. Most of the port terminals have multi-modal hinterland connections. These three modes are also used for the intra-port container distribution. Currently, the port is trying to reduce the share of road transport due to sustainability reasons. In general, the percentage of goods transported by road transport and inland shipping should be reallocated to rail transport. However, specifically for containers, this reallocation from the share of road transport should be reallocated to both the rail transport and inland shipping. Stakeholders from both transport modes have been working collaboratively to improve and optimize the operations of these transport modes.

The author is funded by the Lembaga Pengelola Dana Pendidikan Republik Indonesia (LPDP-RI).

The equipment that are used to load/unload containers from vessels is called *quay cranes* (QCs). The size and specification of a QC may vary, depending on the needs of the terminals. Nowadays, QCs are already designed so that they can handle the biggest container vessels. Apart from the QCs, the port terminals also have smaller cranes that are used specifically to handle the barges. These *barge cranes* (BCs) are not only smaller in size, but also have more variations due to their specific uses. They can be static or mobile.

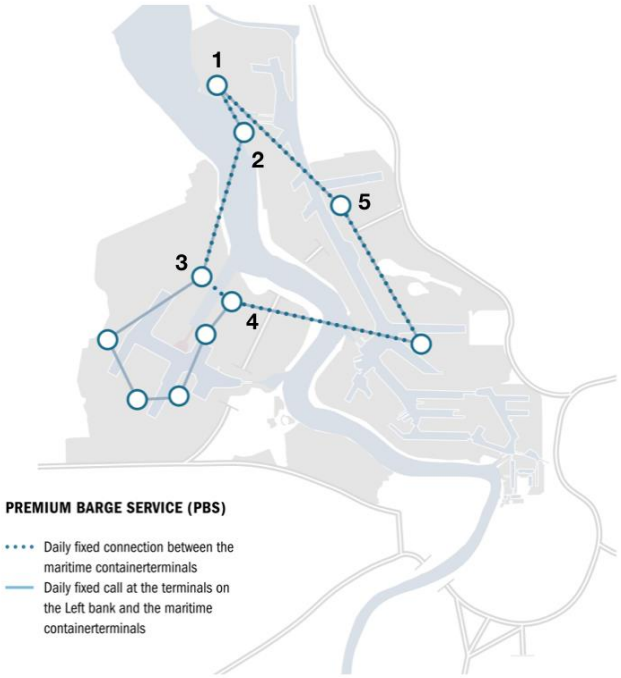


Fig. 1. Five container terminals of Antwerp and the routes of the barge shuttle service [1]

B. Instream: inland shipping initiatives

Instream is a collaborative inland shipping programme between the Port of Antwerp and other close partners initiated to achieve the port's modal split ambition [8]. There are three main concerns being addressed in the programme: a) *nautical coordination*, b) *efficient container handling*, and c) *effective container distribution within the port*.

As an effort to improve the nautical coordination, the *Automatic Identification System* (AIS) is implemented to locate vessels that are sailing around the port. Meanwhile, the *Barge Traffic System* (BTS) allows barge and terminal operators to communicate effectively while allocating time slots of the terminals. A central coordination point is used to coordinate the call schedules of all the port terminals. In order to have less call at the five maritime terminals, the Instream initiatives have introduced a consolidation hub in the port area. Vessels with call sizes smaller than 30 moves can consolidate their volumes at this hub. Moreover, the Instream initiatives also introduce the *Premium Barge Service* (PBS), which is an hour-based barge shuttle service that allows inter terminal container transportation inside the port area. The route of the PBS is shown in Fig. 1. Apart from the five maritime terminals, the PBS also visit quay K364, which is one of the consolidation hubs in the port area.

III. Literature Review

In 2003, Vis and de Koster [2] made an overview of the processes in container terminals. In general, containers arrive at a terminal by the sea-going vessels and the hinterland connections. Sea-going vessels are loaded/unloaded by the QCs. Containers are stored as stacks in the terminal while waiting to be either a) distributed to another terminal by the *Inter Terminal Transport* (ITT) system, b) loaded to a sea-going vessel, or c) delivered to the hinterland by one of the transport modes. This research deals with how to coordinate the logistics operations between a port and an offshore modular platform. Transport between terminals is an essential matter when it comes to how to coordinate the platform's logistics operation to the port's container handling service. Therefore, the ITT will be discussed further in the next section.

A. Inter Terminal Transport

The term ITT was introduced when Ottjes et al. [3] proposed a model to simulate container handling processes in port terminals. The study was done in the year 1996 as a part of the Incomaas project. The study defined the term ITT as *'the transport of containers between terminals, depots and distribution centres in the area of the port with the aid of various transport modes: railways, roads, inland shipping, and sea'*. Since then, the term started to appear in the literature; particularly in those concerning the expansion of the Maasvlakte area in Port of Rotterdam.

Evaluating the performance of a container terminal along with all of its processes is a complicated work, as processes in container terminals have a lot of parameters and variables. Negenborn and Duinkerken [4] proposed a set of parameters that are commonly used in ITT evaluation. Having a top priority to deliver the containers in time, the study of ITT system in Maasvlakte 1&2 proposed the *non-performance rate* as the main performance indicator of an ITT system. The non-performance rate denotes the situation when a container arrived too late at its destination [5]–[8]. Though this indicator was used as the main performance indicator in the project, they also proposed some other indicators to provide more insights on the system performance, such as: a) the occupation rate of both the handling equipment and vehicles, b) the waiting times at the terminals, and c) the average delay of the late containers. Meanwhile, another research [9] used the waiting time and turnaround time of the inland barges as its main performance indicator.

B. ITT evaluation methodologies

Most container terminal problems are evaluated with either *analytical* or *simulation methods.* In their review, Vis and de Koster [2] highlighted some differences between these two methods. The analytical methods interpret the problem as a mathematical model. The model is then assigned with input from a prior data collection. Finally, the solutions are derived from solving the mathematical problem. *Integer programming (IP)*, *queue models*, *network models*, and *assignment problems* are some examples of the analytical methods. Generally, these methods simplify a real-scaled problem in order to reduce the computation time in solving the problem. While these methods help in making decisions at both strategic and tactical levels, it could not represent problems of the operational level due to its simplified features. In contrast, in simulation methods, every process and factors are addressed as detailed as they can be. As a consequence, it will take a long time to construct and validate a simulation model.

Huang et al. [10] compared these two methods in an effort to planning a container terminal. The study stated that the outcomes of the analytical method tend to either underestimate or overestimate the performance of the system. Though, the differences between them are minimal. Nevertheless, both methods are still used in the study of ITT, and research on both sides are still growing. This is observed in the deliverables of the ITT system for Maasvlakte [6]–[8], in which both methods are used to validate the outcomes of the others.

C. Container handling on offshore platform

The concept of a container transshipment terminal on an offshore platform is not something new. Researchers in Japan proposed the *Mega Float Container Terminal Facility* (MFCT) concepts in 2004 to extend the handling capacity of existing container ports. One of the concept is to implement the MFCT as a logistics base in the outer sea area [11]. In 2005, a master student from TU Delft proposed the floating transshipment container terminal concept in his graduation thesis [12]. The thesis consists of operational and financial feasibility studies of different floating terminal concepts and configuration with respect to different hydrodynamical scenarios. The scenarios are constructed from the wind and sea wave conditions at several points in the North Sea. In 2012, Kim and Morrison [13] made a classification of offshore terminal concepts and studied the economic feasibility of these concepts. Later in 2013, a technical and cost evaluation on different concepts of *floating container storage and transhipment terminal* (FCSTT) is presented in [14]. The study reviewed several existing offshore floating terminals, floating storage configurations, and different kinds of floating terminal handling equipment. The study also proposed some design concepts of FCSTT by combining the different configurations and types of equipment. The study opts for the barge-structured platform configuration and suggested to use either rail-mounted slewing cranes or pedestal slewing cranes to handle the incoming container vessels. The proposed concept is based on the application of Gottwald's open-sea floating crane-barge system in Indonesia.

From the operational perspective, both [11] and [12] concluded that the container handling operation of the floating terminal is influenced by the relative motions between the floating terminal and the vessel. Based on [12], the container handling operation on the floating terminal could be actualized with significant sea wave height up to 3 meters. Another study [15] compared the performance of *floating marine terminal* (FMT), or *floaterm*, with the performance of *conventional marine terminal* (CMT) under normal and disruptive conditions. Though, the focus of the study is more to compare the differences between the FMT and CMT, not to show how the FMT relieves the demand for container handling at the CMT.

The dynamics and handling efficiency of a land-based QC has been thoroughly studied in [16] and simulated in [17]. It can be concluded from these studies that the dynamics of a land-based QC rely on the wind conditions. However, the dynamics of a floating QC is not the same. The response of a floating QC to the sea wave motions is evaluated in [18] by taking into account the heave, surge, and sway motions at the tip of the crane's boom. However, since the floating QC is designed for a seaside berth extension of a terminal, the floating QC is assumed to be installed on an individual pontoon-shaped structure rather than a floating terminal. In this study, the heave and surge motions are large, but the time interval between the peaks are also wide. Thus, these two motions can be easily compensated by the operator. Meanwhile, the sway motion is considered to be more difficult to be compensated. In [11], the QC handling efficiency of the MFCT is determined based on a) the reduced acceleration of the grab trolley, and b) the displacement of the container that is being (un)loaded. Both are caused by the oscillation of the MFCT due to the sea wave motions. The study showed that the effect of the oscillation of the MFCT to the QC operation is negligible. On the other hand, Ali et al. [12] concluded that due to the large dimensions of the floating container transhipment terminal, the sea motions would not significantly influence the handling efficiency of the QCs. Moreover, another study has proposed a dedicated control system for mobile/offshore cargo handling designed to reduce the QC's motion responses due to the sea wave motions [19].

The wind conditions and the sea wave motions would also affect the motions of the vessels that are being handled at the floating terminal as well as the sailing speed of the vessels. As indicated in [20], when the sea states are uncompromising, the maximum amount of containers that can be carried by the vessels decreases, resulting in a demand fluctuation along with the seasonal changes within a year. Sukeyasu et al. [11] refer to PIANC to determine the allowable vessel motions in order to be handled by the QCs on the MFCT. Meanwhile, Ali et al. [12] considered the vessels' heave and roll motions to be the most influential modes of motions during the handling process at a floating terminal. Another study [21] evaluated the (un)loading process of a super container vessel from a floating quay during rough weather conditions. In this study, the floating quay is implemented as a seaside berth extension, so vessels are handled in between the floating quay and the landside quay. Thus, a numerical three-body diagram analysis is used to calculate the vessel's motion responses due to the sea wave motions. In order to validate the findings, the numerical analysis result was compared to an experiment result [22]. The study concluded that even though the floating characteristics increases the relative motion response between the vessels and the quay, the response is still within acceptable limits.

Meanwhile, the sailing of the vessels is also affected by the sea states. The effects have been studied since 1998 in [23]. The study constructed a traffic simulation model of the Port of Antwerp waterway. The model was developed by using SIMAN programming language and Arena software. Tides and weather conditions were included as components in the model. Data of the sea wave heights and relevant weather patterns such as strong wind, heavy rain, and fog at multiple points along the waterway were treated as input for the sailing process of the vessels. The vessels can be *tide-dependent* or *-independent*, depending on their size. Tide-dependent is when the sea wave height is higher than the vessel's draught. In this scenario, the vessels have their *tidal window*, which represents the limited period when a vessel can sail during high tides. As the main performance indicator of the model, the service time of the vessels at the port is also size-dependent and classified into five categories. The developed model has been used as a decision tool for expansion and renovation projects at the port area.

IV. Model Development

The previous tasks of the WP9 have proposed several designs and equipment configurations for the S@S platform. The proposed configurations are constructed based on analytical operation research approaches. Predictions on the platform's container handling capacity have been given in terms of its maximum storage capacity and container dwell time. This research will develop a simulation model so that these predictions could be compared to the results of experiments that are conducted with the model. The *discrete-event simulation method* is used in this research. The simulation model will be an object-oriented model at the container level. This means that each of the terminal components is modelled as objects with chains of processes that interact with each other.

The main goal of the model is to evaluate the performance of the Port of Antwerp as well as the S@S platform with respect to different configurations and scenarios. Some criteria have to be fulfilled regarding the model. These criteria are listed below:

a) The model should be able to be used to compare the performance of the port when the platform is installed and not installed,

b) The model should be able to simulate the distribution of containers within the port area as well as between the port and the platform,

c) The model should be able to be used to compare the performance of the platform with different terminal & barge route configuration,

d) The model should be able to generate realistic container handling demand scenarios,

e) The model should be able to incorporate the effect of sea states to the handling capacity of the platform.

Nowadays, there is a lot of simulation software with different functionalities and different programming languages. This research chooses to develop a model under the Python programming language, for the reason that Python is an emerging, free, open-source language; which makes it very popular among software developers. There is also a lot of active users, and the user community runs pretty well. When compared to the Pascal programming language used in [8], Python has more learning materials available on the internet, and it is possible to use data handling software from the extensive open-source libraries. This research uses the *salabim* software package, which is an open-source discrete event simulation software in Python language.

A. *Model boundaries and assumptions*

In the real world, the container facilities of the port (customs, empty depots, repair centres, etc.) are spread along with all the 24 terminals of the port. However, sea-going container vessels can only be handled in the five maritime container terminals. All the containers coming and leaving to/from Port of Antwerp must have been handled in one of these terminals. In addition, vessels that have smaller call sizes must consolidate their volumes in the consolidation hubs. Several studies [9], [24] have been conducted in order to determine where the consolidation hubs should be located in the port area. In these studies, three consolidation hubs are proposed: quay K869, quay K1742, and quay K364. Therefore, this research chooses to include quay K364 in the simulation, and add another terminal as the S@S platform. Moreover, in the real world, the distribution of containers around the port area is done with trucks, trains, and the intra-port barge shuttle service. However, this research assumes that the distribution is done solely by the intra-port barge shuttle service and fixed sailing time between one terminal to another. In the simulation, the barges are assumed to operate 24/7, while in the real world the barges only operate 18 hours per day.

The containers are distinguished as import and export containers. Import containers are carried by the sea-going vessels, and they need to be delivered to a certain terminal. Meanwhile, export containers are the containers coming from the hinterland that need to be loaded onto the vessels. Each container has its due time. Due time of the import containers can be +1, +2, or +3 days after their arrival time. Concurrently, due time of the export containers is the departure time of the vessels they should be loaded onto. Some other assumptions regarding the simulation model are listed as follow:

a) A vessel visits either the S@S platform or one of the terminals of the Port of Antwerp.
b) There is no direct transhipment service at the platform. In this way, containers that are handled at the platform must go to one of the terminals of the port before being delivered to the hinterland.
c) Quay length of the terminals is not taken into account.
d) Once a vessel is assigned a certain number of QCs, other QCs that are just available from their previous vessel cannot be assigned to the same vessel.
e) The barge shuttle service is handled by a dedicated BC in each terminal.
f) The barge shuttle's sailing time between the terminals is fixed.
g) The traffic of the Scheldt river and wind condition around the port area are considered to be constantly fine.
h) All the equipment and the barges operate 24/7 with no downtimes due to maintenance, shift changes, etc.

Fig. 2 illustrates the input and output of the simulation model. The input for the simulation includes 1) the terminal configurations, 2) container handling demand, 3) route strategies for the barge shuttle, and 4) the sea states scenarios. On the other hand, the output of the simulation is the performance indicators of the system. This research uses several indicators in order to compare the performance of the port-platform system when the platform is installed and not installed. These indicators are as follow:

1) *QC utilization rate* — the ratio of occupied QCs to the total number of QCs in the terminals.
2) *Vessels & containers handled* — the number of vessels and containers that are handled at the terminals.
3) *Statistics of the terminals' stacks* — the mean and max. number of TEUs stored in the terminals' stacks
4) *Vessel handling time* — the amount of time vessels spends in the terminals.
5) *Container dwell time* — the amount of time containers spends in the terminals.
6) *Non-performance rate* — when containers are delivered later than their due times, they are registered as non-performance.

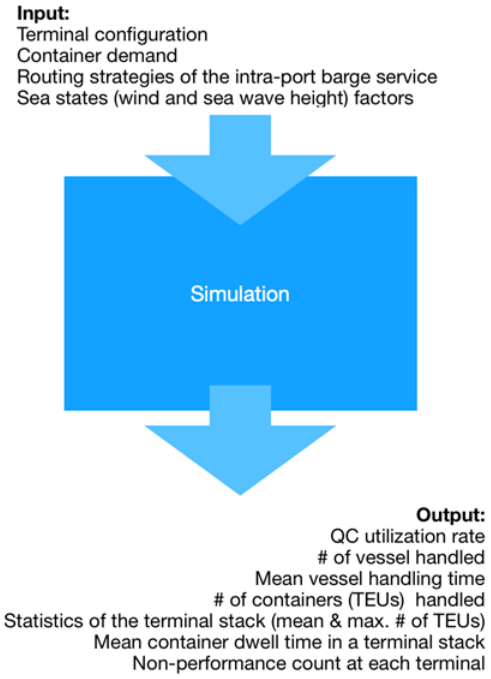


Fig. 2. 'Black box' diagram of the model

Schematic of the model is shown in Fig. 3. There are eight objects within the simulation model: a) *VesselGenerator*, b) *Vessel*, c) *Container*, d) *Terminal*, e) *Barge*, f) *QuayCrane*, g) *BargeCrane*, and h) *Weather*. All objects are active, except the Container and Terminal.

B. *Allocation of QCs to the vessels*

The number of QC that can be assigned to the vessels depends on the vessel's capacity. Table 1 shows the relation of the vessel's capacity and the maximum number of QCs that can be assigned to the vessels.

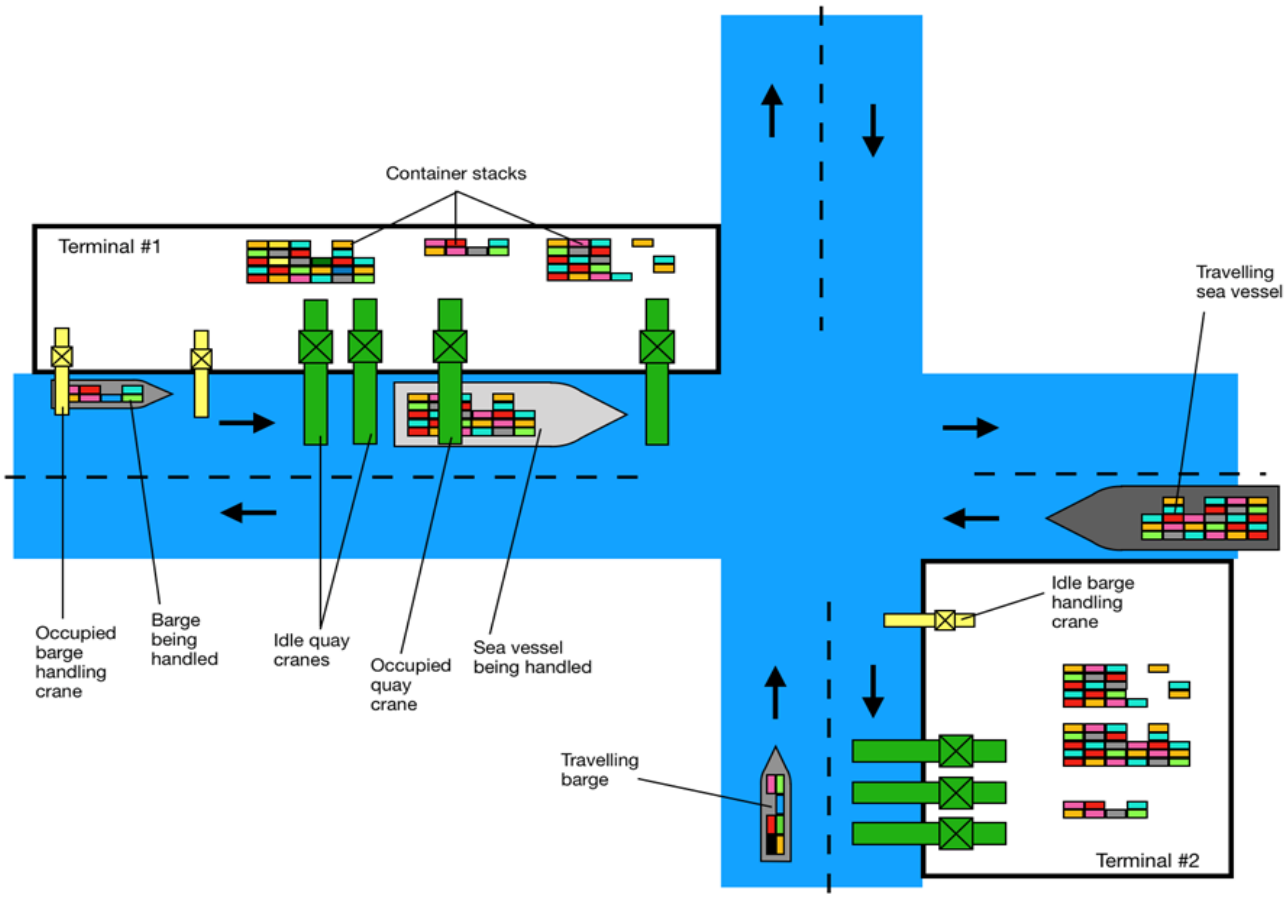


Fig. 3. Schematic diagram of the model

TABLE I RELATION OF VESSEL'S CAPACITY AND MAXIMUM NUMBER OF QCS [25], [26]

Vessel capacity	Max. number of QCs
≤ 800 TEUs	3
800 TEUs ≤ x ≤ 2500 TEUs	5
2500 TEUs ≤ x ≤ 4500 TEUs	7
4500 TEUs ≤ x ≤ 8000 TEUs	9
≥ 8000 TEUs	12

C. *Vessel allocation algorithm*

The offshore modular platform is assumed to be implemented as an extension for the other terminals in terms of handling containers. Vessels will prefer to call at the port terminals when one of the port terminals is less utilized than the platform. An algorithm is proposed to allocate vessels to the terminals. The algorithm is based on three parameters: a) vessel's capacity, b) storage capacity of the platform, and c) terminal's QC utilization rate. The decision tree is illustrated in Fig. 4.

D. *Demand distributions*

In the simulation model, the vessels and containers are generated based on several distributions. These distributions are a) the interarrival distribution, b) the vessel capacity distribution, and c) the vessel's call size distributions.

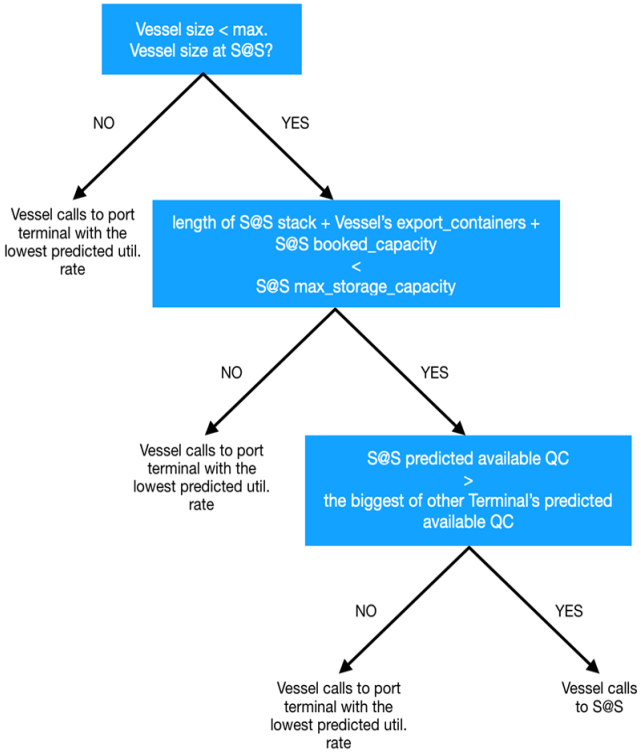


Fig. 4. Decision tree for the vessel allocation algorithm

The specification of the distributions is determined based on the number of vessels that visit the port in a one-year period. The vessel's interarrival time in the model is sampled from a normal distribution $N(\mu, \sigma^2)$ with mean μ = 92 minutes and standard deviation σ^2 = 20 minutes. Meanwhile, the vessel's capacity and call sizes are sampled from cumulative probability distributions which are shown in Table II and Table III, respectively.

TABLE II DISTRIBUTION OF THE VESSEL'S CAPACITY

Vessel capacity	Cumulative distribution (%)
≤ 100 TEUs	0
≤ 1,500 TEUs	36.4
≤ 3,000 TEUs	53.4
≤ 4,500 TEUs	66
≤ 6,000 TEUs	76.1
≤ 10,500 TEUs	90.8
≤ 21,000 TEUs	100

TABLE III DISTRIBUTION OF VESSEL'S CALL SIZES

Number of containers	Export call size distribution (%)	Import call size distribution (%)
≤ 100 TEUs	0	0
≤ 500 TEUs	30	30
≤ 1,000 TEUs	60	60
≤ 1,500 TEUs	75	80
≤ 2,000 TEUs	85	90
≤ 2,500 TEUs	90	95
≤ 3,000 TEUs	95	98
≤ 6,000 TEUs	100	100

E. Barge route strategies

This research evaluates two barge route strategies. In the first strategy, the barge shuttle visits each terminal once in a day. When the platform is present, the barge shuttle also visits the platform. In the second strategy, two barge shuttle routes are used. The first route is a loop route that goes to the terminals in the port area, while the second route is a dedicated connection between the platform and one of the port terminals. These strategiess are illustrated in Fig. 5 – Fig. 7.

F. Sea states scenario

A simple sea states scenario is constructed based on a historical dataset of the wind speed and significant sea wave height at the location of the platform. These sea states are assumed to affect the move time of QCs at the platform as well as the barge's sailing time. The mathematical relations between the sea states and these two variables are given as follow:

$$QC's\ move\ time = \frac{move\_time}{wind\ factor \times sea\ wave\ factor} \quad (1)$$

$$sailing\ time_{port-platform} = \frac{4\ hours}{sea\ wave\ factor} \quad (2)$$

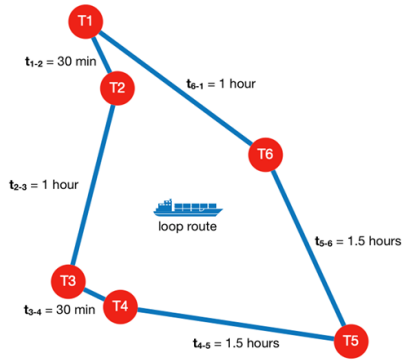


Fig. 5. A loop route strategy without the platform

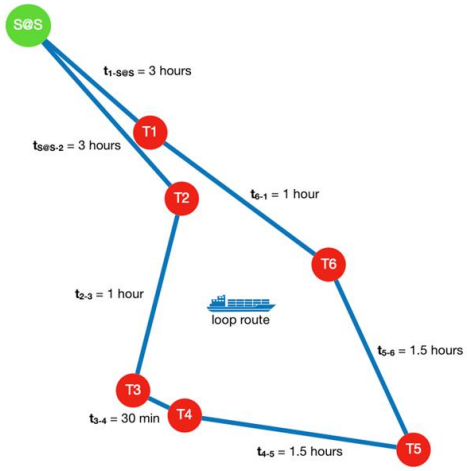


Fig. 6. A loop route strategy with the platform

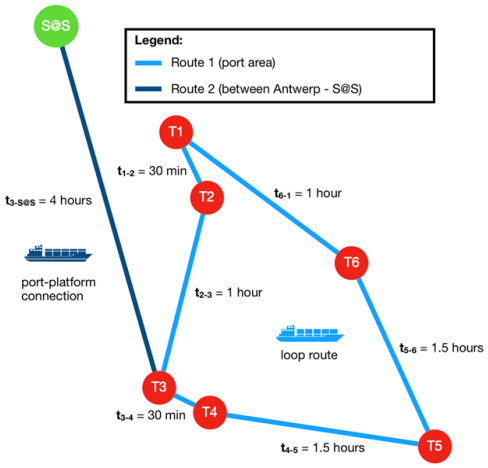


Fig. 7. Two barge shuttle routes strategy

In order to reduce the number of variables, the values in the dataset are categorized. This categorization is shown in Table IV.

TABLE IV SEA STATES CATEGORIZATION [12], [16], [27]

Wind speed categories	**Efficiency factor (%)**
$0 \text{ m/s} < v_{wind} \leq 10 \text{ m/s}$	100
$10 \text{ m/s} < v_{wind} \leq 20 \text{ m/s}$	75
$20 \text{ m/s} < v_{wind} \leq 30 \text{ m/s}$	50
Sea wave height categories	**Efficiency factor (%)**
$0 \text{ m} < h_{sea\text{-}wave} \leq 1.5 \text{ m}$	100
$1.5 \text{ m} < h_{sea\text{-}wave} \leq 3 \text{ m}$	50
$h_{sea\text{-}wave} > 3 \text{ m}$	25

G. Model verification

The verification of a model checks if the model behaves as it is specified. The *salabim* software package allows the traces of the simulation to be monitored. Also, an animation window has been made to visualize the processes of the simulation. The simulation model has been verified by conducting a series of test runs and checks using these two features.

V. EXPERIMENTS & RESULTS

In order to investigate how the performance of the port is affected by the presence of the platform, five cases are constructed with different barge route and vessel allocation strategies. These configurations are shown in Table V.

TABLE V EXPERIMENTAL PLAN

Experiment #	**Barge route strategy**	**Vessel allocation strategy**
1 (ref.)	Fig. 5 (platform not present)	-
2	Fig. 6	≤ 6,000 TEUs
3	Fig. 6	≥ 6,000 TEUs
4	Fig. 7	≤ 6,000 TEUs
5	Fig. 7	≥ 6,000 TEUs

Each of the cases is simulated for 60,480 minutes (6 weeks) in the simulation world. This is equivalent to 2-4 hours in real time; depending on the complexity of the configurations and scenarios.

Case #1 shows the performance of the system when the platform is not present. The result of this case is compared to the available historical data and the result of similar studies in order to validate the model. In case #2-#5, the platform is present. The results of these cases are compared to those of case #1 to see how the port's handling capacity is affected by the presence of the platform with different strategies.

Furthermore, a preferred strategy is selected from these five cases for each of the barge route strategy and the vessel allocation strategy. These strategies are used to conduct sensitivity analysis to provide additional insights on how the platform would perform with respect to other configurations and scenarios, such as: a) different number of QCs on the platform, b) different locations of the platform, c) different distributions of the vessel's inter-arrival time and call sizes, and d) different sea states scenarios.

A. Model validation

The validation phase of a model checks if the model is suitable to represent the real problem. A model would never be completely the same as the real system. Therefore, it is important to indicate to what extent that the model outcomes can be considered 'valid'.

Table VI shows the result of case #1. This experiment result is expected to replicate the performance of the current real-world configuration. It is observed from the result that the overall mean QC utilization rate of the port is 47.5%, which is lower than the expected value (70-90%). This is caused by the assumption that QCs that have finished handling a vessel cannot be reallocated to another vessel that has started to be handled by other QCs. Therefore, in the experiment, it is possible that a vessel is only handled by two QCs even though there are other available QCs in the terminal. Despite, this value corresponds with a report which stated that the utilization rate of the port is relatively low compared to those of the other ports [28]. At the same time, the values for the vessel handling time is always lower than one day, and this corresponds with the value mentioned in [29].

TABLE VI PORT PERORMANCE IN EXPERIMENT #1

Performance indicator	**Values**
Mean QC utilization rate	47.38%
Mean vessel handling time	10.46 hours
Max. number of TEUs in avg.	12708 TEUs
Mean container dwell time	1.66 days

The Port of Antwerp claims an overall QC productivity of 40 crane moves per hour per crane. This means that in the real world a QC takes about 1-2 minutes to load/unload a container to/from a vessel. However, if the QuayCrane's and BargeCrane's *move_time* in the model are set to 1.5 minutes, the non-performance rate of the model will reach about 70-80% of the total containers that are generated during the simulation time. This high value of non-performance rate is not only caused by the *move_time* values but also due to two other reasons: a) the assumption that the container distribution is done solely by the barge shuttle service, and b) the vessels and containers are handled only based on *first in, first out* (FIFO) algorithm. In order to have a lower non-performance rate, there needs to be a dedicated planning and control scheme to allocate vessels based on their departure times and to sort containers based on their due times. However, this planning and control scheme is out of the scope of this research. Therefore, in order to compensate the need

for the planning and control scheme, this research recommends setting the QuayCrane's and BargeCrane's *move_time* to lower values. Though, it is important to note that lowering the QuayCrane's and BargeCrane's *move_time* would also affect the statistics of the terminal stacks. If the *move_time* value is too low, the containers will leave the terminal stack too quickly, and the number of containers that are stored in the terminals and their dwell time values would become unrealistic.

It is also possible to compensate the high non-performance rate by adding more barge shuttles which operate only in the port area. In this way, the container distribution between the port terminals will be enhanced, and the non-performance error will be localized to the S@S platform. Though, this non-performance localization is not relevant for case #2 and case #3, due to the implemented barge route strategy. The non-performance rate of case #1 reaches a value close to zero if there are 12 barge shuttles operating in the port area. With the same amount of barge shuttles that operate in the port area, the non-performance rate of case #4 decreases from about 55% to 5%. However, adding more barge shuttles will significantly increase the computational load of the simulation.

For these reasons, this research chooses to compensate the high non-performance rate value by setting the QuayCrane's *move_time* to 0.5 minute, and the BargeCrane's *move_time* to 0.1 minute. The QuayCrane's *move_time* is equivalent to a QC productivity rate of 120 TEUs/hour. On the other hand, the low value of the BargeCrane's *move_time* can be considered as having four BCs at the terminals to handle the barge shuttle at once. Though these values seems too optimistic, they provide a trade-off point between the high non-performance rate and the validity of the statistics of the terminal stack. With these configurations, the statistics of the terminal stacks in case #2 and #4 can be compared to the Space@Sea memorandum document. In this document, the dwell time of containers in the port terminals is assumed to be between 3-5 days, while the dwell time on the platform is assumed to be 1-3 days. These assumptions are used to calculate predictions of the platform's storage capacity with respect to different demand scenarios. In the simulations, though, the container dwell time is not an assumption, but a measure of the port performance. The results of the simulations show that the mean container dwell time at the port terminal ranges between 1.5-2 days, while the mean container dwell time on the platform ranges between 0.5-1 days. Despite the difference, both the memorandum document and the simulation results show that the container dwell time on the platform is always half of the dwell time at the port terminal. Moreover, with the same demand scenario, it is also observed from the simulation results that the maximum number of containers on the platform is also half of the memorandum's predicted storage capacity of the platform.

B. Port performance with different strategies

Even though there are 612 vessels that are generated within the 6-week period, only 577 vessels have arrived and handled at one of the terminals. In addition, the number of containers that have been handled by the system ranges between 850,000-900,000 TEUs. The installation of the platform will allocate some of the vessels to be handled at the platform. When the platform is set to handle vessels with capacity ≤ 6,000 TEUs (case #2 and #4), around 15% of the total vessels are handled at the platform. When the platform is set to handle vessels with larger capacity (case #3 and #5), only 7.5-8% of the total vessels are handled at the platform. Concurrently, the installation of the platform will also allocate some of the containers to be handled at the platform. The percentages of containers that are handled at the platform ranges from 7.5-9.5%. Case #2 has the highest percentage (9.28%), followed by case #5 (7.86%). The percentages of case #3 and #4 are 7.35% and 9.03%, respectively. The statistics of the platform stack (mean, maximum, and dwell time of the containers) reach the highest values in case #2 and the lowest values in case #5.

Comparison of the port's performance indicators with respect to the different strategies are illustrated in Fig. 8 – Fig. 11. In general, the installation of the platform reduces the average QC utilization rate of the port terminals. In case #2 and #3, the maximum number of containers in the port terminals increases by 7.7% and 4.5%, respectively; while in case #4 and #5, this number decreases by 0.94% and 6.3%, respectively. Therefore, it can be concluded that the installation of the platform with the two-barge-route configuration will reduce the port's utilization rate and required storage capacity.

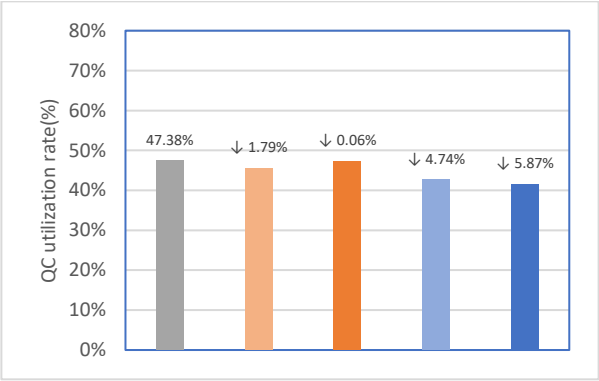


Fig. 8. Average QC utilization rate of the port terminals

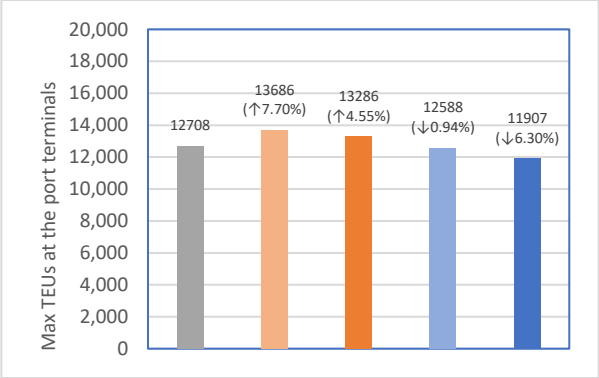


Fig. 9. Average max. number of TEUs at the port terminals

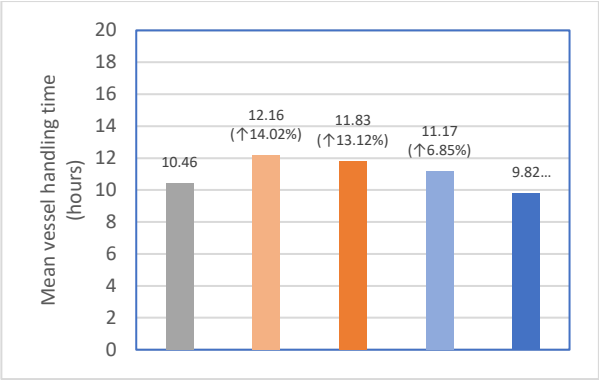


Fig. 10. Mean vessel handling time at the port terminals

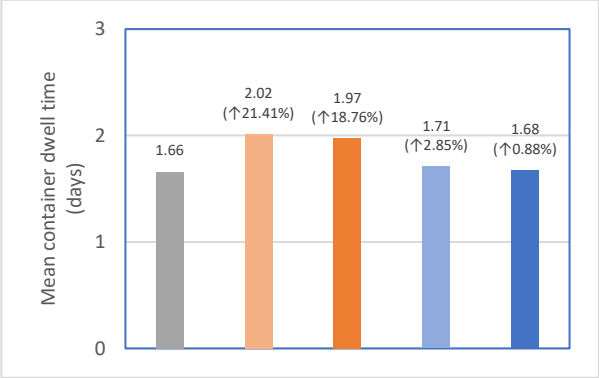
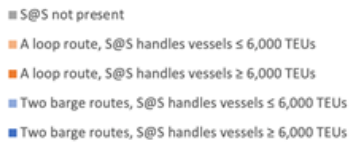


Fig. 11. Mean container dwell time at the port terminals



Except for case #5, the mean vessel handling time at the port terminals always increase when the platform is present. The increases become more significant when the platform only handle vessels ≤ 6,000 TEUs. This is because a portion of the small vessels is handled at the platform. Therefore, the average vessel size that visit the port terminal increases. Larger vessels tend to have larger call sizes. The vessels have pre-determined departure time which is based on their call sizes, so it is likely for the large vessels to stay longer at the terminals while they are being handled. However, this parameter is also affected by the container distribution service, as vessels are set to be able to leave before their pre-determined departure time if all their containers have been handled. This is observed in case #3: even though the large vessels are handled at the platform, the mean vessel handling time at the port terminals still increases. This is due to the low distribution service in case #3. Meanwhile, the mean container dwell time always increase when the platform is present; because there are more export containers that have to wait in terminal #3 before being transferred to the platform.

C. Platform performance with different strategies

Fig. 12 – Fig. 15 show the comparison of the platform performance with the different strategies. In general, with the same vessel allocation strategy, the two barge routes strategy provides lower QC utilization rate than the single route strategy. Moreover, with the same barge route strategy, the platform's QC utilization rate is always lower when the platform only handles vessels ≤ 6,000 TEUs. The mean vessel handling time at the platform shows a similar trend with the QC utilization rate.

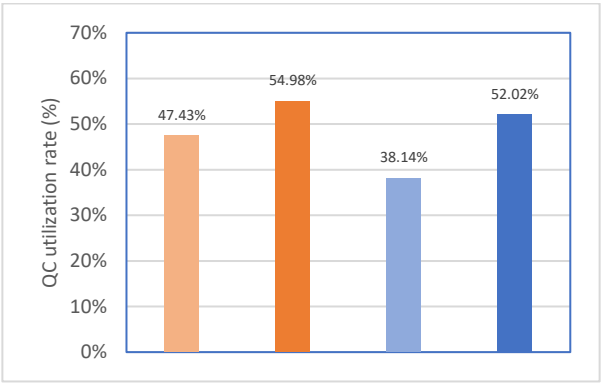


Fig. 12. QC utilization rate of the platform

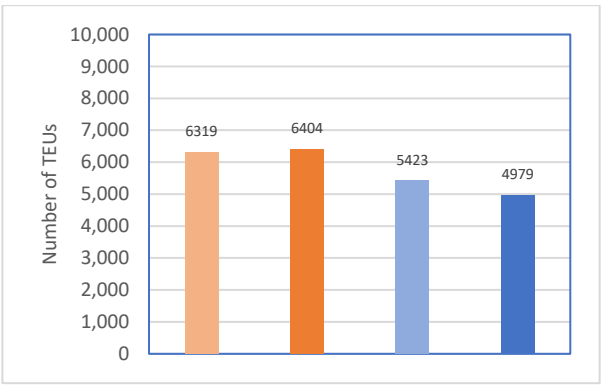


Fig. 13. Maximum number of TEUs on the platform

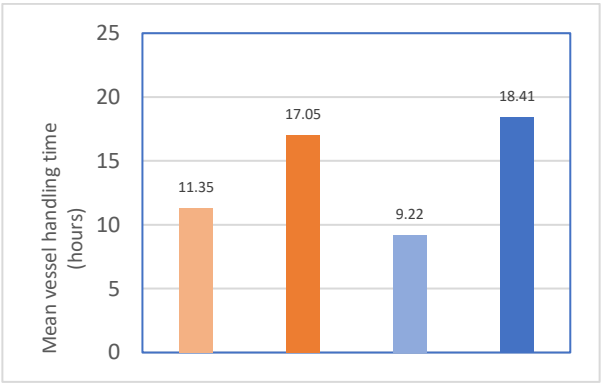


Fig. 14. Mean vessel handling time at the platform

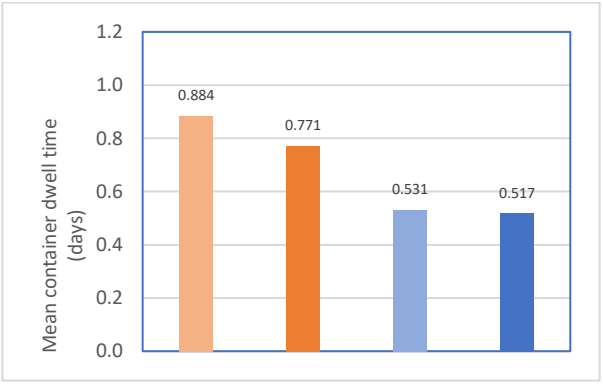


Fig. 15. Mean container dwell time at the platform

The main purpose of the S@S platform is to reallocate some of the handling demand at the port to the platform. It is preferred that the platform can handle more demand with less QC and storage space utilization. In addition, even though the platform installation increases the overall container handling time of the port, it is still preferred that the containers that are handled at the platform should be delivered on time. In this case, a lower non-performance rate is preferred. Therefore, this research suggests implementing the strategies from case #4 for the following reasons:

1) It provides the highest percentage of vessels that are handled at the platform (14.54%) and the second-highest percentage of containers that are handled at the platform (9.03%). The highest container percentage is in case #2. However, case #2 has a higher non-performance when compared to case #4.
2) As shown in Fig. 12 – Fig. 15, case #4 provides a better S@S platform performance. Case #5 might have a lower maximum number and dwell time of containers that are stored on the platform, but it also has a lower percentage of containers that are handled at the platform.

To have more insight on the platform's performance with respect to other configurations, the following sections will discuss several sensitivity analyses that are conducted using the strategies from case #4.

D. Different number of QCs at the platform

Four simulations are conducted with different number of QCs at the platform. The results are shown in Table VII. The 10-crane configuration is used as a reference. Even when the mean QC utilization rate decreases by 3.63%, the 15-crane configuration increases the percentage of vessels and containers that are handled at the platform by 2.5% and 2% respectively. Though not significant, the mean vessel handling time also increases. This might be because more vessels are handled at the platform. The reductions in the platform's share to the non-performance and the maximum number of containers that are stored on the platform indicate that adding more QCs allows more export containers to be loaded onto their vessels on time.

TABLE VII SENSITIVITY ANALYSIS ON THE NO. OF QUAY CRANES AT THE PLATFORM

Performance indicators	**Number of quay cranes**			
	2	**5**	**10 (ref)**	**15**
Platform's share to the non-performance rate	2% *(+0.89%)*	1.8% *(+0.69%)*	1.11%	1% *(-0.11%)*
Mean QC utilization rate	65.40% *(+27.26%)*	58.67% *(+20.53%)*	38.14%	34.51% *(-3.63%)*
Vessels handled	72 *(-2.5%)*	76 *(-2%)*	89	101 *(+2.5%)*
Mean vessel handling time (h)	12.73 *(+38.07%)*	11.74 *(+27.33%)*	9.22	9.38 *(+1.74%)*
Containers handled	78281 *(-0.73%)*	79519 *(-0.6%)*	85188	100157 *(+2%)*
Max. no. of TEUs on the platform	3272 *(-39.66%)*	4819 *(-11.13%)*	5423	4588 *(-15.39%)*
Mean container dwell time (days)	0.483 *(-9.03%)*	0.485 *(-8.66%)*	0.531	0.539 *(+1.51%)*

In contrast, the percentage of vessels that are handled at the platform decreases by 2% in the 5-crane configuration and by 2.5% in the 2-crane configuration. The percentage of containers that are handled at the platform decrease by 0.6% in the 5-crane configuration and by 0.73% in the 2-crane configuration. At the same time, the QC utilization rate increases by 20.53% in the 5-crane configuration and by 27.26% in the 2-crane configuration. Moreover, reducing the number of QCs at the platform increases the non-performance rate by about 1%, and the mean vessel handling time at the platform by 2.5-3.5 hours. Meanwhile, the mean container dwell time does not change significantly with the different QC configuration.

E. Different platform locations

Apart from the pre-determined location, two other potential locations of the platform are examined. The first potential location is the mouth of the Scheldt river. It is assumed that the sailing time between the port and this particular location is 2 hours. The other location is the border of the offshore operations, at which it is assumed that the barge shuttle should sail for 6 hours from the port to reach the platform. The results of the experiments are shown in Table VIII. It is observed from the table that closer location reduces the platform's share to the non-performance as well as the maximum number and dwell time of containers that are stored on the platform; and vice versa. However, the changes are not significant.

TABLE VIII SENSITIVITY ANALYSIS ON THE PLATFORM LOCATIONS

Performance indicators	**Sailing time to the platform**		
	2 hours	**4 hours (ref)**	**6 hours**
Platform's share to the non-performance rate	0.88% *(-0.23%)*	1.11%	1.8% *(+0.69%)*
Mean QC utilization rate	41.12% *(+2.98%)*	38.14%	47.16% *(+9.02%)*
Vessels handled	87 *(-0.33%)*	89	85 *(-0.65%)*
Mean vessel handling time (h)	9.73 *(+5.53%)*	9.22	11.84 *(+28.42%)*
Containers handled	86467 *(+0.14%)*	85188	99871 *(+1.56%)*
Max. no. of TEUs on the platform	5200 *(-4.11%)*	5423	5951 *(+9.74%)*
Mean container dwell time (days)	0.495 *(-6.78%)*	0.531	0.602 *(+13.37%)*

Though the location of the platform is not taken into account in the vessel allocation algorithm, the different number of vessels that are handled at the platform in each experiment indicates that there is an effect of the platform location to the allocation of the vessels. The platform location directly affects the barge operation, which plays an important role in the container distribution between the port and the platform. At the same time, the container distribution affects

the QC occupancy and availability of the terminals. Therefore, the platform location indirectly affects the QC occupancy and availability, which are taken into account in the allocation algorithm. Due to the same reason, there are also no trends observed in the mean QC utilization rate, mean vessel handling time, as well as the number of containers that are handled at the platform.

F. Different demand scenarios

In the future, the demand for container handling at the Port of Antwerp will increase. Therefore, this research will make an assumption in order to construct some future container handling demand scenarios. The Antwerp Port Authority does not include any demand projection in their annual publication, so this research will use the same projections as the ITT studies for Maasvlakte 1&2. There are four future economic trends considered in the ITT studies. These trends are: a) *Low Growth*, b) *European Trend*, c) *Global Economy*, and d) *High Oil Price*. As presented in Table IX, each of these trends predicts the number of containers to be handled at the Port of Rotterdam in the year 2030. The demand growth factors are the ratio of the baseline (year 2008) to the respective future trend. The Global Economy trend has the highest factor, so this research will use the value 2.76.

TABLE IX FUTURE DEMAND PROJECTIONS

Future demand projections	Demand in Rotterdam (10^6 tonnes/year)	Growth factor
Baseline (2008)	112.30	1.00
2030 Low-Growth	190.00	1.69
2030 European Trend	267.00	2.38
2030 Global Economy	310.00	**2.76 (chosen)**

One can safely assume that the demand growth factor from the year 2008 to the year 2017 is 1.15. Based on this assumption, the growth factor from the year 2017 to 2030 can be calculated as follow:

$$growth\ factor_{2017-2030} = \frac{1}{1.15} \times 2.76 = 2.4 \quad (3)$$

This value is used to construct several future demand scenarios that are evaluated with the simulation model. The demand scenarios include the following: a) increase of the vessel's interarrival time, b) increase of the vessel's call sizes.

1. Increase of vessels' interarrival time

In this scenario, the growth factor is accommodated in the interarrival time of the vessels. It is also assumed that the distribution of the interarrival time becomes wider. This assumption is accommodated by increasing the standard deviation. So, the interarrival time of this experiment is set to a non-negative normal distribution $N(\mu, \sigma^2)$ with mean $\mu = 40\ minutes$ and standard deviation $\sigma^2 = 40\ minutes$. With the new interarrival time, there are 1150 vessels and 1.7 million TEUs generated during the 6-week period. The comparison of the result to those of the reference configuration ($\mu = 100\ minutes,\ \sigma^2 = 20\ minutes$) is given in Table X.

TABLE X SENSITIVITY ANALYSIS ON THE VESSEL'S INTERARRIVAL TIME

Performance indicators	Interarrival time distribution	
	$\mu = 40\ minutes,$ $\sigma^2 = 40\ minutes$	$\mu = 100\ minutes,$ $\sigma^2 = 20\ minutes$ **(ref.)**
Platform's share to the non-performance rate	2.14% *(+1.03%)*	1.11%
Mean QC utilization rate	67.99% *(+29.85%)*	38.14%
Vessels handled	186 *(+1.63%)*	89
Mean vessel handling time (h)	9.89 *(+7.27%)*	9.22
Containers handled	146764 *(-0.4%)*	85188
Max. no. of TEUs on the platform	6202 *(+14.36%)*	5423
Mean container dwell time (days)	0.681 *(+28.25%)*	0.531

In general, the new of interarrival time doubles the platform's share to the non-performance rate and escalates the QC utilization rate by about 30%. The statistics of the platform's stack also increase. However, the percentages of vessels and containers that are handled at the platform do not change significantly.

2. Increase of vessels' call sizes

In this scenario, the growth factor is assumed to affect the call size of the vessels. This is done by multiplying each of the bin classes in the call size distributions (Table III) by the growth factor. If the distribution of the vessels' capacity (Table II) is not changed along with the call size distribution, the increase of the call sizes will be bounded by the vessel capacity. In this way, the number of containers that are generated within the simulation period would not reach the predicted value. Therefore, to accommodate the increase of the call sizes, the vessel capacity distribution is also shifted by multiplying the mean of this distribution by the same growth factor. However, the upper bound for the capacity distribution is set to stay the same (21,000 TEUs). In addition, two other simulations are conducted with the same increase in the call sizes, but different vessel allocation strategy. In these extra simulations, the platform is set to only handle ULCS.

With these modifications, there are 604 vessels and 1,823,901 TEUs that are generated within the simulation period. The comparison of these scenarios to the reference scenario is shown in Table XI. The increase of the vessel's call sizes causes a rise in both the number of vessels and containers that are handled at the platform. However, the percentages decrease by about 2%. With this demand scenario, the mean QC utilization rate and the mean vessel handling time at the platform also increase significantly. These are the effects of having the same number of vessels with doubled call sizes. It is also observed that the platform's share to the non-performance rate slightly decreases.

TABLE XI SENSITIVITY ANALYSIS ON THE VESSELS' CALL SIZES & PLATFORM'S VESSEL ALLOCATION STRATEGY

Performance indicators	Call size scenarios & vessel allocation strategy			
	2017 demand, ≤ 6,000 TEUs only (ref.)	**2030 demand, ≤ 6,000 TEUs only**	**2017 demand, ULCS only**	**2030 demand, ULCS only**
Platform's share to the non-performance rate	1.11%	0.99% *(-0.12%)*	1.15% *(+0.04%)*	1.52% *(0.41%)*
Mean QC utilization rate	38.14%	66.52% *(+28.38%)*	40.55% *(+2.41%)*	76.39% *(+38.25%)*
Vessels handled	89	102 *(+2.12%)*	30 *(-9.64%)*	38 *(-8.33%)*
Mean vessel handling time (h)	9.22	15.83 *(+71.69%)*	15.81 *(+71.47%)*	31.99 *(+246%)*
Containers handled	85188	125502 *(-2.15%)*	47118 *(-4.04%)*	104738 *(-3.28%)*
Max. no. of TEUs on the platform	5423	6916 *(+27.53%)*	4853 *(-10.51%)*	6952 *(+28.19%)*
Mean container dwell time (days)	0.531	0.681 *(+28.24%)*	0.667 *(+25.61%)*	0.772 *(+45.39%)*

At the same time, in both 2017 and 2030 demand scenarios, when the S@S platform only handles ULCS, the percentages of the vessels and containers that are handled at the platform decrease. This is due to two reasons: a) there are only about 5% of ULCS from the total number of vessels, and b) the ULCS do not necessarily have large call sizes. Moreover, compared to when the platform handles vessels ≤ 6,000 TEUs, the platform's share to the non-performance rate is always higher. This could mean that most of the export containers that are associated with the vessels which are handled at the platform are not loaded to the vessels on time.

G. Different platform's maximum storage capacity

This research assumes that the allocation of the vessels that visit the port-platform system takes into account the maximum storage capacity of the platform. In this way, when the platform's capacity is fully occupied (or booked), vessels would not be allocated to the platform. A capacity of 10,000 TEUs is used as the reference value for the platform's maximum storage capacity. Some simulations are conducted with both 2017 and 2030 demand scenarios to see the effect of the different value of maximum storage capacity to the performance of the platform.

Table XII shows the results of three simulations with 2017 demand scenario and different platform's maximum storage capacity. Though not significant, fewer vessels and containers are handled at the platform when the platform's maximum storage capacity is set to 5,000 TEUs. This configuration also increases the mean QC utilization rate and mean vessel handling time at the platform. However, the dwell time of the containers decreases. On the other hand, the platform's performance does not change when the maximum storage capacity is set to 20,000 TEUs. This indicates that with the 2017 demand scenario, the allocation of the vessels are not constrained by the platform's storage capacity when it is set to 10,000 TEUs.

TABLE XII SENSITIVITY ANALYSIS ON THE PLATFORM'S MAXIMUM STORAGE CAPACITY WITH 2017 DEMAND SCENARIOS

Performance indicators	Platform's maximum storage capacity		
	5,000 TEUs	**10,000 TEUs (ref)**	**20,000 TEUs**
Platform's share to the non-performance rate	1.13% *(+0.02%)*	1.11%	1.11% *(0%)*
Mean QC utilization rate	41.64% *(+3.5%)*	38.14%	38.14% *(0%)*
Vessels handled	87 *(-0.33%)*	89	89 *(0%)*
Mean vessel handling time (h)	9.8 *(+6.29%)*	9.22	9.22 *(0%)*
Containers handled	84583 (-0.06%)	85188	85188 *(0%)*
Max. no. of TEUs on the platform	5423 *(0%)*	5423	5423 *(0%)*
Mean container dwell time (days)	0.497 *(-6.4%)*	0.531	0.531 *(0%)*

In contrast, Table XIII shows that in the simulations with 2030 demand scenarios, a higher value of the platform's maximum storage capacity shifts the performance of the platform. When the platform's capacity is set to 10,000 TEUs, some vessels that have large call sizes choose to not to call to the platform because the platform's capacity would not suffice their call sizes. It can be seen that the platform handles 0.15% more containers when the storage capacity is doubled to 20,000 TEUs. When the number of containers that are handled at the platform increases, more containers have to wait before being transferred to the port. Therefore, the dwell time of the containers and the platform's share to the non-performance also increase.

TABLE XIII SENSITIVITY ANALYSIS ON THE PLATFORM'S MAXIMUM STORAGE CAPACITY WITH 2030 DEMAND SCENARIOS

Performance indicators	Platform's maximum storage capacity	
	10,000 TEUs (ref)	**20,000 TEUs**
Platform's share to the non-performance rate	0.99%	2.31% *(+1.32%)*
Mean QC utilization rate	66.52%	63.93% *(-2.59%)*
Vessels handled	102	102 *(0%)*
Mean vessel handling time (h)	15.83	16.56 *(+4.61%)*
Containers handled	125502	122850 *(-0.15%)*
Max. no. of TEUs on the platform	6916	5767 *(-19.92%)*
Mean container dwell time (days)	0.681	1.07 *(+57.12%)*

Though the percentage of vessels that are handled at the platform does not change, it is indicated from the increase of the vessel handling time that the call sizes of the vessels that visit the platform increase when the storage capacity is set to 20,000 TEUs.

H. Different sea states scenarios

To have insights on how the sea states affect the handling capacity of the platform, three simulations are conducted with different sea states scenarios. The scenario that have been constructed (Section IV.F) is used as the reference scenario. The other two scenarios are extreme scenarios. One scenario assumes that the sea states are always in bad conditions. Meanwhile, the other scenario assumes that the sea states are always fine. The comparison between these three sea states scenarios is shown in Table XIV. Unfortunately, there is no available information on how an offshore terminal would perform with respect to the scenarios. Thus, it is impossible to validate the result of these experiments.

TABLE XIV SENSITIVITY ANALYSIS ON THE SEA STATES SCENARIOS

Performance indicators	Sea states scenarios		
	Bad sea states	**Real sea states scenario (ref)**	**Fine sea states**
Platform's share to the non-performance rate	8.49% *(+7.38%)*	1.11%	0.77% *(-0.34%)*
Mean QC utilization rate	44.93% *(+6.79%)*	38.14%	40.73% *(+2.59%)*
Vessels handled	92 *(+0.49%)*	89	84 *(-0.82%)*
Mean vessel handling time (h)	10.57 *(+14.64%)*	9.22	9.94 *(+7.81%)*
Containers handled	64953 *(-2.14%)*	85188	90558 *(+0.57%)*
Max. no. of TEUs on the platform	7626 *(+40.62%)*	5423	4321 *(-20.32%)*
Mean container dwell time (days)	1.468 *(+176%)*	0.531	0.400 *(-24.67%)*

Nevertheless, the table shows that the platform performs best in the fine sea states scenario in terms of the number of containers that are handled and stored at the platform. In this scenario, the platform handles 9.6% of the total generated containers. This percentage is 0.57% higher than the percentage in the experiment with real sea-state scenario. Due to the constantly fine sea states, the containers spend less than half a day in average on the platform, and the maximum number of containers that are stored on the platform does not reach 5,000 TEUs. The platform's share to the non-performance rate is also lowest in this scenario. Moreover, the sea states directly affect the crane's and barge shuttle's operation. In this way, these states influenced the crane occupancy and the number of containers that are currently stored on the platform, which both are used in the vessel allocation algorithm. This explains why in the fine scenario the platform handles less vessels in spite of the increased container percentage.

VI. CONCLUSION & FUTURE RECOMMENDATIONS

In general, the installation of the platform will increase the total handling capacity of the port. Some vessels and containers will be handled at the platform, and this will reduce the mean QC utilization rate and the need for storage capacity at the port terminals. Moreover, when the platform is installed, some sea-going vessels would not have to sail to the port to have their containers handled. In this way, the vessels that are handled at the platform could save about 8 hours of sailing time to/from the port. This might also reduce the vessel traffic at the Scheldt river. However, as it is assumed that there is no direct transhipment to the hinterland terminals, all the containers that are handled at the platform should be transferred to the port before being delivered to their hinterland destinations. As a consequence, the platform installation will increase the non-performance rate of the system.

This research has developed a model to evaluate the performance of a port when its container distribution service should be extended by the presence of an offshore modular platform. This tool can be used by the S@S project partners to evaluate other configurations and scenarios that have not been described in this research. Based on the experiment results, this research suggests implementing the two-barge-route configuration as the container distribution scheme between the port and the platform. This research also recommends the platform to handle vessels with capacity ≤ 6,000 TEUs. Moreover, even though this research focuses on the Port of Antwerp as the case study, this model can be used for the evaluation of other port by modifying the terminal and barge route configurations.

The model development phase is constrained by the time limitation of this research. Therefore, the level of details that are given in this model is also limited. In order to have a more valid model, it is recommended for future research to use the model and expand the program code by introducing other important parameters for the vessel allocation algorithm. For instance, instead of only the available QCs, the vessel can also decide where to call by considering the quay length of the terminal, or by taking into account the weather forecast at the location of the platform. Moreover, while there is a random characteristic in the generated demand, this research only used the FIFO algorithm to handle the incoming vessels and containers. There is a huge probability that the vessels are not fully handled, and the containers are delivered later than their due times. The model will generate more valid outcomes if better planning and control are introduced to allocate the vessels based on their departure times and to sort the containers based on their due times. It is also suggested to integrate this model with a traffic modelling of the Scheldt river, so the operations of the barge shuttle can be evaluated more accurately.

In reality, a direct transhipment scheme is more preferred for the offshore modular platform, as barges from the hinterland can directly visit the platform instead of having to go to the port. In this way, the installation of the platform will have a more significant impact on the overall logistics operations. Therefore, this research suggests future work to investigate the extension of the coordination with the hinterland terminals. Furthermore, a cost-benefit analysis can be conducted to have insights on the economic aspects of the platform installation.

ACKNOWLEDGMENT

This paper is a documentation of the graduation project titled '*Performance evaluation tool for the expansion of a port's container network by an offshore modular platform'*, which is one of the requirement to complete the author's

Masters of Science study in the track of Transport Engineering and Logistics at Delft University of Technology. This graduation project is conducted as a part of the task T9.5 under the Work Package 9 (WP9) of the Space@Sea project.

The author would like to grant a special gratitude to Prof. Rudy Negenborn as the chair of the author's graduation committee. Special appreciations should be addressed to Mark Duinkerken and Dimitris Souravlias who have supervised and helped the author during this project; and Dingena Schott as the WP-9 leader and one of the author's graduation committee. The author also appreciate all the critics, and supports, and small talks from the people of Maritime & Transport Technology Department of TU Delft who are involved in the Space@Sea project (Pieter, Giannis, Bilge, Vittorio).

REFERENCES

[1] Antwerp Port Authority, "Instream: Smart and efficient inland navigation," Antwerp, 2015.

[2] I. F. A. Vis and R. De Koster, "Transshipment of containers at a container terminal: An overview," *Eur. J. Oper. Res.*, vol. 147, no. 1, pp. 1–16, 2003.

[3] J. A. Ottjes, M. B. Duinkerken, J. J. M. Evers, and R. Dekker, "Robotised Inter Terminal Transport of containers: A simulation study at the Rotterdam Port Area," in *Proceedings of the 8th European Simulation Symposium*, 1996.

[4] R. R. Negenborn and M. B. Duinkerken, "Definition of common parameter values required for ITT system design," Delft, 2014.

[5] E. J. Gerritse, "Analysis for Inter Terminal Transportation demand scenarios for the Maasvlakte I and II in 2030," Delft, 2014.

[6] K. Tierney, S. Voß, and R. Stahlbock, "A mathematical model of inter-terminal transportation," *Eur. J. Oper. Res.*, vol. 235, no. 2, pp. 448–460, 2014.

[7] F. Nieuwkoop, F. Corman, R. R. Negenborn, M. B. Duinkerken, M. Van Schuylenburg, and G. Lodewijks, "Decision support for vehicle configuration determination in Inter Terminal Transport system design," *Proc. 11th IEEE Int. Conf. Networking, Sens. Control. ICNSC 2014*, pp. 613–618, 2014.

[8] H. J. L. Schroer, F. Corman, M. B. Duinkerken, R. R. Negenborn, and G. Lodewijks, "Evaluation of inter terminal transport configurations at Rotterdam Maasvlakte using discrete event simulation," in *Proceedings of the 2014 Winter Simulation Conference*, 2014, pp. 1771–1782.

[9] A. Caris, C. Macharis, and G. K. Janssens, "Network analysis of container barge transport in the port of Antwerp by means of simulation," *J. Transp. Geogr.*, vol. 19, no. 1, pp. 125–133, 2011.

[10] W.-C. Huang, T.-C. Kuo, and S.-C. Wu, "A comparison of analytical methods and simulation for container terminal planning," *J. Chinese Inst. Ind. Eng.*, vol. 24, no. 3, pp. 200–209, 2007.

[11] Y. Sukeyasu *et al.*, "New proposal of evaluation method for cargo handling efficiency on mega float container terminal facility," *Ocean. '04 Mts/IEEE Techno-Ocean '04, Vols 1- 2, Conf. Proceedings, Vols. 1-4*, pp. 1067–1072, 2004.

[12] A. Ali and H. Ligteringen, "Floating Transshipment Container Terminal," Delft University of Technology, 2005.

[13] J. Kim and J. R. Morrison, "Offshore port service concepts: Classification and economic feasibility," *Flex. Serv. Manuf. J.*, vol. 24, no. 3, pp. 214–245, 2012.

[14] A. J. Baird and D. Rother, "Technical and economic evaluation of the floating container storage and transhipment terminal (FCSTT)," *Transp. Res. Part C Emerg. Technol.*, vol. 30, pp. 178–192, 2013.

[15] M. A. Dulebenets, M. M. Golias, S. Mishra, and W. C. Heaslet, "Evaluation of the floaterm concept at marine container terminals via simulation," *Simul. Model. Pract. Theory*, vol. 54, pp. 19–35, 2015.

[16] H. Schim van der Loeff, "Quay Crane productivity at the ECT Delta terminal," Delft University of Technology, 2007.

[17] P. Chhetri, G. B. Jayatilleke, V. O. Gekara, A. Manzoni, and B. Corbitt, "Container terminal operations simulator (CTOS) – Simulating the impact of extreme weather events on port operation," *EJTIR*, no. 16, pp. 195–213, 2016.

[18] B. J. A. Pielage, J. C. Rijsenbrij, and H. Ligteringen, "Floating cranes for container handling," *2008 1st Int. Conf. Infrastruct. Syst. Serv. Build. Networks a Bright. Futur. INFRA 2008*, 2008.

[19] E. H. Kim *et al.*, "An advanced cargo handling system operating at sea," *Int. J. Control. Autom. Syst.*, vol. 12, no. 4, pp. 852–860, 2014.

[20] A. A. Shabayek and W. W. Yeung, "Effect of Seasonal Factors on Performance of Container Terminals," *J. Waterw. Port, Coastal, Ocean Eng.*, vol. 127, no. 3, pp. 135–140, 2001.

[21] M. H. Kim, B. Kumar, and J. W. Chae, "Performance Evaluation of Loading/Offloading from Floating Quay to Super Container Ship," in *Proceedings of the Sixteenth International Offshore and Polar Engineering COnference*, 2006, vol. 4, pp. 144–149.

[22] H. Y. Kang, M. H. Kim, J. H. Kim, W. S. Park, and J. W. Chae, "Offloading From Both Sides of a Super-container Ship to Land and Floating Harbor Predicted vs . Experimental Results," in *Proceedings of the Twentieth International Offshore and Polar Engineering Conference*, 2010, vol. 7, pp. 581–587.

[23] G. F. Thiers and G. K. Janssens, "A Port Simulation Model as a Permanent Decision Instrument," *Simulation*, vol. 71, no. 2, pp. 117–125, 1998.

[24] K. Braekers, A. Caris, and G. K. Janssens, "Optimal shipping routes and vessel size for intermodal barge transport with empty container repositioning," *Comput. Ind.*, vol. 64, no. 2, pp. 155–164, 2013.

[25] "How much bigger can container ships get? - BBC News." [Online]. Available: https://www.bbc.com/news/magazine-21432226. [Accessed: 27-Jan-2019].

[26] J. Martin, S. Martin, and S. Pettit, "Container ship size and the implications on port call workload," *Int. J. Shipp. Transp. Logist.*, vol. 7, no. 5, p. 553, 2015.

[27] Nederlands Normalisatie-instituut, "NEN 2018: Cranes - Loads and combinations of loads," Delft, 1983.

[28] Ports Regulator of South Africa, "Port Benchmarking Report 2015/2016," Durban, 2016.

[29] C. Ducruet and O. Merk, "Examining container vessel turnaround times across the world," *Port Technol. Int.*, pp. 18–20, 2013.

Appendix B Map of the port and modal split comparison to Rotterdam

Available at: https://www.portofantwerp.com/sites/portofantwerp/files/POA-2059_Publiekskaart_EN_500101_LR.pdf [Accessed: 27-Aug-2018]

Figure B.1 Enlarged version of the port map

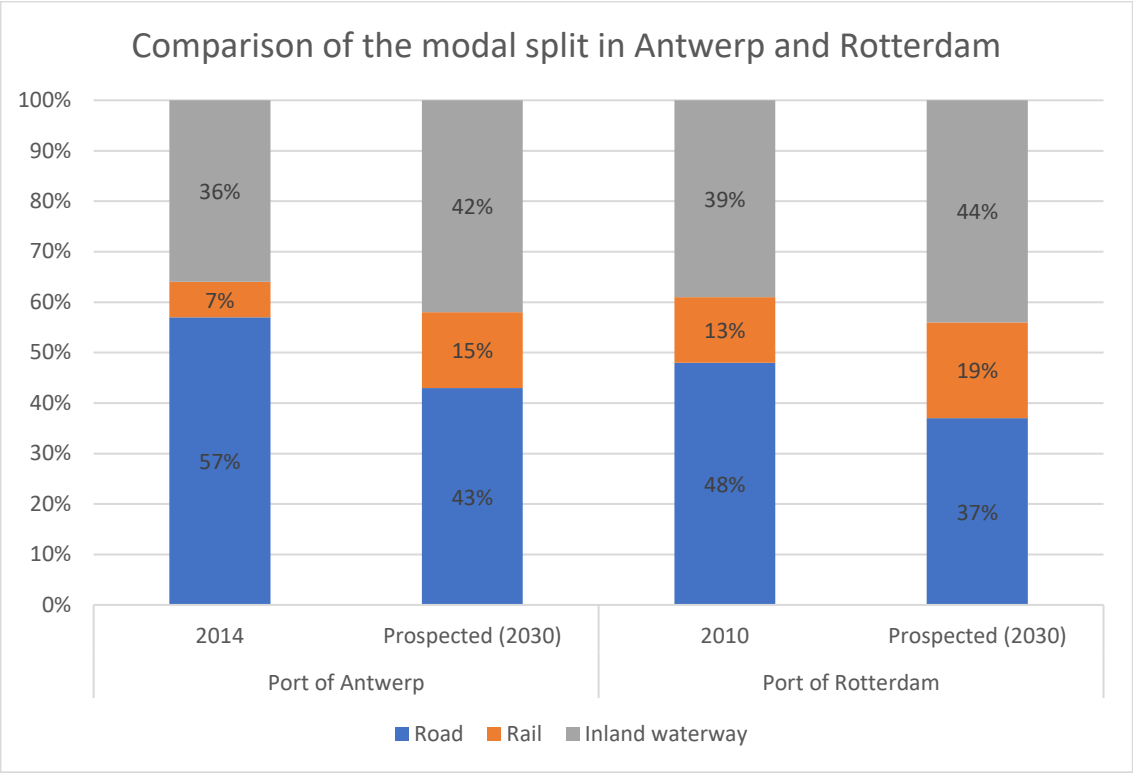


Figure B.2 Comparison of the modal split between Antwerp and Rotterdam [14], [35]

Appendix C Documents related to T9.4 and T9.5

(These documents have been removed from this report version due to confidentiality reasons)

Appendix D Input and output files

```
# input file for terminal specification
12 #noofOC
|
2 #noofBC
```

Figure D.1 Screenshoot of input file ' Terminal[ID#].txt'

```
# input file for premium barge service antwerp

#terminal number - call size (min) - sail time to next terminal (min)

Terminal1 120 30
Terminal2 60 60
Terminal3 120 30
Terminal4 180 150
Terminal5 120 90
Terminal6 120 60
```

Figure D.2 Screenshoot of input file 'case1.txt'

```
#factor wind # factor wave
1       1
1       1
1       1
1       1
1       1
1       1
1       1
1       1
1       1
1       1
1       1
1       1
1       1
1       1
1       1
1       1
1       1
1       1
1       1
1       0.5
1       0.5
1       0.5
1       0.5
1       0.5
1       0.5
1       0.5
1       0.5
1       0.5
1       0.5
1       1
```

Figure D.3 Screenshoot of input file 'inputweather.txt'

```
Total vessel generated: 612 vessels

Total containers generated: 942826 TEUs

Non-performance rate: 51.64 %

================================
Terminal 1:
================================

QC Utilization rate: 43.37 %

Vessel handled: 83 vessels

Mean vessel handling time: 9.85 hours

Stack statistics:

 - Total handled: 161232 TEUs, On-time: 59029 TEUs, Late (import): 75367 (45383) TEUs

 - Mean dwell time: 1.829 days

Statistics of Terminal1 stack at     60480
                                                                  all    excl.zero         zero
---------------------------------------- -------------- ------------ ------------ ------------
Length of Terminal1 stack                duration           60480        60480            0
                                         mean                7373.337     7373.337
                                         std.deviation       2093.513     2093.513

                                         minimum                0            0
                                         median              7615         7615
                                         90% percentile      9637         9637
                                         95% percentile     10074        10074
                                         maximum            13362        13362

Length of stay in Terminal1 stack        entries           161232       161232            0
                                         mean                2634.166     2634.166
                                         std.deviation       1461.079     1461.079

                                         minimum                0.100        0.100
                                         median              2991.275     2991.275
                                         90% percentile      4325.183     4325.183
                                         95% percentile      4636.078     4636.078
                                         maximum             5510.830     5510.830
```

Figure D.4 Screenshoot of output file

Appendix E Code of the model

```
import salabim as sim
import sys
import numpy as np
import pandas as pd
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
import seaborn as sns
import datetime
# import random
# from random import randrange


class Container(sim.Component):

    pass  # the Containers are passive objects, they have no process

    def animation_objects(self):
        destination_color = {1: 'blue', 2: 'red', 3: 'green', 4: 'yellow',
5: 'white', 6: 'orange', 7: 'black'}
        ao0 = sim.AnimateRectangle((0, 5, 5, 0),
fillcolor=destination_color[self.destination], arg=self)
        return 7, 7, ao0


class VesselGenerator(sim.Component):

    def process(self):
        while True:
            # create vessel
            Vessel()

            # inter-arrival time
            yield self.hold(sim.Bounded(sim.Normal(100, 20), 0)())


class Vessel(sim.Component):

    def setup(self):
        self.import_load = sim.Queue(self.name())
        self.export_load = sim.Queue(self.name())

        # vessel capacity distribution
        self.capacity = int(sim.Cdf((100, 0, 1500, 40, 3000, 50, 4500, 70,
6000, 80, 10000, 90, 21000, 100)).sample())

        # determines the number of export containers
        self.export_containers = int(
```

```
sim.Bounded(sim.Cdf((100,0,500,30,1000,60,1500,75,2000,85,2500,90,3000,95,
6000,100)), lowerbound=100,
                        upperbound=self.capacity)())

        # self.export_containers = int(
        #     sim.Bounded(sim.Normal(0.52 * self.capacity, 0.25 *
self.capacity), lowerbound=100,
        #                 upperbound=self.capacity)())

        # determines the number of import containers
        self.import_containers = int(

sim.Bounded(sim.Cdf((100,0,500,30,1000,60,1500,80,2000,90,2500,95,3000,98,
6000,100)), lowerbound=100,
                        upperbound=self.capacity)())

        # self.import_containers = int(
        #     sim.Bounded(sim.Normal(0.48 * self.capacity, 0.25 *
self.capacity), lowerbound=100,
        #                 upperbound=self.capacity)())

        self.arrival_time = env._now + env.days(sim.IntUniform(1,
3).sample())
        self.departure_time = self.arrival_time + (
                env.minutes(0.5) * (self.export_containers +
self.import_containers))

        # determines the maximum number of quay crane with respect to the
Vessel capacity
        if self.capacity <= 1500:
            self.max_quaycrane = 8
        elif 1500 < self.capacity <= 3000:
            self.max_quaycrane = 10
        else:
            self.max_quaycrane = 12

    def animation_objects(self):
        ao1 = sim.AnimatePolygon((0, 0, 10, 0, 10, 10), fillcolor='red',
arg=self)
        return 15, 15, ao1

    def process(self):
        print("{0:.2f}".format(env._now)+' vessel created with capacity '+
"{0:.0f}".format(
                    self.capacity)+' TEUs, '+
"{0:.0f}".format(self.import_containers) +' import '+ "{0:.0f}".format(
                    self.export_containers)+' export')
        # to determine utilization rate at port area
        utilization_rate_at_port = [(len(Terminal1.quaycraneQ) -
Terminal1.booked_quaycrane),
                                    (len(Terminal2.quaycraneQ) -
Terminal2.booked_quaycrane),
                                    (len(Terminal3.quaycraneQ) -
Terminal3.booked_quaycrane),
                                    (len(Terminal4.quaycraneQ) -
Terminal4.booked_quaycrane),
                                    (len(Terminal5.quaycraneQ) -
Terminal5.booked_quaycrane),
```

```
                                    (len(Terminal6.quaycraneQ) -
Terminal6.booked_quaycrane)]

        # vessel decide where to call:
        if self.capacity <= 3000:
            if len(SpaceAtSeaPlatform.stack) + self.export_containers +\
                    SpaceAtSeaPlatform.booked_capacity <=
SpaceAtSeaPlatform.max_storage_capacity:
                if max(utilization_rate_at_port) <= (
                        len(SpaceAtSeaPlatform.quaycraneQ) -
SpaceAtSeaPlatform.booked_quaycrane):
                    self.destination = 7
                    SpaceAtSeaPlatform.booked_capacity +=
self.export_containers
                else:
                    self.destination =
utilization_rate_at_port.index(max(utilization_rate_at_port)) + 1
            else:
                self.destination =
utilization_rate_at_port.index(max(utilization_rate_at_port)) + 1
        else:
            self.destination =
utilization_rate_at_port.index(max(utilization_rate_at_port)) + 1
        vessel_destination[self.destination].upcoming_call += 1
        vessel_destination[self.destination].booked_quaycrane +=
self.max_quaycrane

        #creating containers associated with the vessel
        for i in range(self.import_containers):
            self.container = Container().enter(self.import_load)
            self.container.origin = 0
            self.container.vessel = self
            self.container.due_time = self.arrival_time +
env.days(sim.IntUniform(1, 3).sample())
            self.container.destination = sim.IntUniform(1, 6).sample()
            # if self.container.due_time - self.arrival_time >= 2880:
            #     self.container.modality = 'barge'
            # else:
            #     self.container.modality = 'truck/train'
        for i in range(self.export_containers):
            self.container = Container()
            self.container.origin = sim.IntUniform(1, 6).sample()
            self.container.vessel = self
            self.container.due_time = self.departure_time
            self.container.destination = self.destination
            # if self.container.due_time - env._now >= 2880:
            #     self.container.modality = 'barge'
            # else:
            #     self.container.modality = 'truck/train'
            self.container.enter(container_origin[self.container.origin])

        env.containers_generated += (self.import_containers +
self.export_containers)
        self.terminal = vessel_destination[self.destination]
        incoming_from_hinterland.trigger()
        yield self.hold(till=self.arrival_time)
        self.terminal.upcoming_call -= 1
        self.terminal.booked_quaycrane -= self.max_quaycrane
        if self.terminal == SpaceAtSeaPlatform:
```

```
            SpaceAtSeaPlatform.booked_capacity -= self.export_containers
        self.enter(self.terminal.vesselQ)
        while len(self.terminal.quaycraneQ) == 0 and self.departure_time >
env._now:
            yield self.wait(self.terminal.quaycrane_available,
fail_at=self.departure_time)

        #vessel bail
        if len(self.terminal.quaycraneQ) == 0:
            self.leave(self.terminal.vesselQ)
            env.non_performance_count += len(self.import_load)
            monitor_non_performance_rate.tally(env.non_performance_count *
100 / env.containers_generated)
            self.terminal.late_delivery += len(self.import_load)
            self.terminal.late_import += len(self.import_load)
            print("{0:.2f}".format(env._now)+' non-performance registered
by vessel while bailing: '+ "{0:.2f}".format(
                    env.non_performance_count * 100 /
env.containers_generated)+' %')
        else:
            for i in range(self.max_quaycrane):
                for QuayCrane in self.terminal.quaycraneQ:
                    QuayCrane.leave(self.terminal.quaycraneQ)
                    QuayCrane.vessel = self
                    if QuayCrane.ispassive():
                        QuayCrane.activate()
                        break


class Terminal(sim.Component):

    pass

    def setup(self):
        self.quaycraneQ = sim.Queue(self.name() + ' quay crane Q')
        self.bargecraneQ = sim.Queue(self.name() + ' barge crane Q')
        self.vesselQ = sim.Queue(self.name() + ' vessel Q')
        self.bargeQ = sim.Queue(self.name())
        self.stack = sim.Queue(self.name() + ' stack')
        self.unloaded_from_barge = sim.State('unloaded from barge')
        self.unloaded_from_vessel = sim.State('unloaded from vessel')
        self.quaycrane_available = sim.State('QC available')
        self.specification = sim.ItemFile(self.name() + '.txt')
        self.id = terminal_index[self.name()]
        if self.name() == 'SpaceAtSeaPlatform':
            self.booked_capacity = 0
            self.max_storage_capacity = 10000
        self.noofquaycrane = self.specification.read_item_int()
        self.noofstraddlecarrier = self.specification.read_item_int()
        self.noofbargecrane = self.specification.read_item_int()
        self.upcoming_call = 0
        self.booked_quaycrane = 0
        self.late_delivery = 0
        self.late_import = 0
        self.ontime_delivery = 0
        for i in range(self.noofquaycrane):
            self.quaycrane = QuayCrane().enter(self.quaycraneQ)
            self.quaycrane.terminal = self
        for i in range(self.noofbargecrane):
```

```
            self.bargecrane = BargeCrane().enter(self.bargecraneQ)
            self.bargecrane.terminal = self


class Barge(sim.Component):

    def animation_objects(self):
        pbs_color = {'PBS1': 'blue',
                     'PBS2': 'red',
                     'PBS3': 'green'}
        ao5 = sim.AnimateCircle(radius=5,
fillcolor=pbs_color[self.name()], arg=self)
        return 12, 7, ao5

    def setup(self):
        self.my_route = sim.ItemFile(PBS_route[self.name()])
        self.terminal = Terminal
        self.bargecrane = BargeCrane
        self.load = sim.Queue(self.name())

    def process(self):
        while True:
            # determine which terminal to moor at
            try:
                self.terminal = eval(self.my_route.read_item())
            except EOFError:
                self.my_route = sim.ItemFile(PBS_route[self.name()])
                self.terminal = eval(self.my_route.read_item())

            # moor at terminal
            self.bargecrane = self.terminal.bargecraneQ.head()
            self.bargecrane.barge = self
            self.bargecrane.leave(self.terminal.bargecraneQ)
            self.bargecrane.activate()
            self.enter(self.terminal.bargeQ)
            yield self.hold(self.my_route.read_item_int())

            #leave terminal
            self.bargecrane.barge = None
            self.bargecrane.passivate()
            self.bargecrane.enter(self.terminal.bargecraneQ)
            self.bargecrane = None
            self.leave(self.terminal.bargeQ)
            self.terminal = None

            #sail to next terminal
            yield self.hold(self.my_route.read_item_int())


class Weather(sim.Component):

    def setup(self):
        self.weather_input = sim.ItemFile('inputweather.txt')
        self.wind_efficiency_at_platform =
self.weather_input.read_item_float()
        self.wave_efficiency_at_platform =
self.weather_input.read_item_float()

    def process(self):
```

```
        while True:
            try:
                self.wind_efficiency_at_platform =
self.weather_input.read_item_float()
                self.wave_efficiency_at_platform =
self.weather_input.read_item_float()
            except EOFError:
                self.weather_input = sim.ItemFile('inputweather.txt')
                self.wind_efficiency_at_platform =
self.weather_input.read_item_float()
                self.wave_efficiency_at_platform =
self.weather_input.read_item_float()
            yield self.hold(60)


class QuayCrane(sim.Component):

    def setup(self):
        self.container = Container
        self.move_time = 0.5
        self.vessel = None

    def animation_objects(self):
        if self.ispassive():
            ao2 = sim.AnimateCircle(radius=5, fillcolor='green', arg=self)
            return 12, 7, ao2
        if self.iswaiting():
            ao2 = sim.AnimateCircle(radius=5, fillcolor='yellow',
arg=self)
            return 12, 7, ao2
        else:
            ao2 = sim.AnimateCircle(radius=5, fillcolor='red', arg=self)
            return 12, 7, ao2

    def process(self):
        while True:
            while self.vessel == None:
                yield self.passivate()

            # unload
            while len(self.vessel.import_load) != 0 and
self.vessel.departure_time > env._now:
                for Container in self.vessel.import_load:
                    if Container.destination == self.terminal.id:
                        Container.leave(self.vessel.import_load)
                        if Container.due_time < env._now:
                            env.non_performance_count += 1
                            monitor_non_performance_rate.tally(
                                env.non_performance_count * 100 /
env.containers_generated)
                            self.terminal.late_delivery += 1
                            self.terminal.late_import += 1
                            print("{0:.2f}".format(env._now)+' non-
performance registered by QC while unloading: ' + "{0:.2f}".format(
                                        env.non_performance_count * 100 /
env.containers_generated) + ' %')
                        else:
                            self.terminal.ontime_delivery += 1
                    else:
```

```
                        Container.leave(self.vessel.import_load)
                        Container.enter(self.terminal.stack)
                        self.terminal.unloaded_from_vessel.trigger()
                    break
                if self.terminal == SpaceAtSeaPlatform:
                    yield self.hold(
                        self.move_time / (
                                    Weather.wind_efficiency_at_platform *
Weather.wave_efficiency_at_platform))
                else:
                    yield self.hold(self.move_time)

            # register the import non performance of the leaving vessel
            if len(self.vessel.import_load) != 0:
                env.non_performance_count += len(self.vessel.import_load)

monitor_non_performance_rate.tally(env.non_performance_count * 100 /
env.containers_generated)
                self.terminal.late_delivery +=
len(self.vessel.import_load)
                self.terminal.late_import += len(self.vessel.import_load)
                for Container in self.vessel.import_load:
                    Container.leave(self.vessel.import_load)
                print("{0:.2f}".format(env._now)+' non-performance
registered by QC, vessel is not fully unloaded: ' + "{0:.2f}".format(
                        env.non_performance_count * 100 /
env.containers_generated) + ' %')

            # load
            while len(self.vessel.export_load) !=
self.vessel.export_containers and self.vessel.departure_time > env._now:
                for Container in self.terminal.stack:
                    if Container.destination == self.terminal.id and
Container.vessel == self.vessel:
                        Container.leave(self.terminal.stack)
                        Container.enter(self.vessel.export_load)
                        self.terminal.ontime_delivery += 1
                        break
                else:
                    yield self.wait(self.terminal.unloaded_from_barge,
fail_at=self.vessel.departure_time)
                if self.terminal == SpaceAtSeaPlatform:
                    yield self.hold(self.move_time / (
                            Weather.wind_efficiency_at_platform *
Weather.wave_efficiency_at_platform))
                else:
                    yield self.hold(self.move_time)

            # register non-performance
            for Container in self.terminal.stack:
                if Container.due_time < env._now:
                    Container.leave(self.terminal.stack)
                    env.non_performance_count += 1

monitor_non_performance_rate.tally(env.non_performance_count * 100 /
env.containers_generated)
                    self.terminal.late_delivery += 1
                    if Container.origin == 0:
                        self.terminal.late_import += 1
```

```
                        print("{0:.2f}".format(env._now)+' non-performance
registered by QC, import containers not leaving origin terminal: ' +
"{0:.2f}".format(
                            env.non_performance_count * 100 /
env.containers_generated) + ' %')
                    else:
                        print("{0:.2f}".format(env._now)+' non-performance
registered by QC, vessel is not fully loaded: ' + "{0:.2f}".format(
                            env.non_performance_count * 100 /
env.containers_generated) + ' %')

            if self.vessel in self.terminal.vesselQ:
                self.vessel.leave(self.terminal.vesselQ)
            self.vessel = None
            self.enter(self.terminal.quaycraneQ)
            self.terminal.quaycrane_available.trigger()


class BargeCrane(sim.Component):

    def setup(self):
        self.container = Container
        self.barge = None
        self.move_time = 0.1

    def animation_objects(self):
        if self.ispassive():
            ao4 = sim.AnimateCircle(radius=5, fillcolor='green', arg=self)
            return 12, 7, ao4
        if self.isstandby():
            ao4 = sim.AnimateCircle(radius=5, fillcolor='yellow',
arg=self)
            return 12, 7, ao4
        else:
            ao4 = sim.AnimateCircle(radius=5, fillcolor='red', arg=self)
            return 12, 7, ao4

    def process(self):
        while True:
            while self.barge == None:
                yield self.passivate()
            if self.terminal == SpaceAtSeaPlatform:
                yield self.hold(
                    self.move_time / (Weather.wind_efficiency_at_platform
* Weather.wave_efficiency_at_platform))
            else:
                yield self.hold(self.move_time)

            #unload

            ##activate for case 3 only##
            if self.barge.name() == 'PBS2' and self.terminal == Terminal3:
                for Container in self.barge.load:
                    if Container.origin == 0 and Container.destination ==
self.terminal.id:
                        Container.leave(self.barge.load)
                        if Container.due_time < env._now:
                            env.non_performance_count += 1
                            monitor_non_performance_rate.tally(
```

```
                                env.non_performance_count * 100 /
env.containers_generated)
                            self.terminal.late_delivery += 1
                            self.terminal.late_import += 1
                            print("{0:.2f}".format(env._now)+' non-
performance registered by BC, import containers late arrival: ' +
"{0:.2f}".format(
                                    env.non_performance_count * 100 /
env.containers_generated) + ' %')
                        else:
                            self.terminal.ontime_delivery += 1
                    elif Container.origin == 0 and Container.destination
!= self.terminal.id:
                        Container.leave(self.barge.load)
                        Container.enter(self.terminal.stack)
                        self.terminal.unloaded_from_barge.trigger()
            elif (self.barge.name() == 'PBS1' or self.barge.name() ==
'PBS3') and self.terminal == Terminal3:
                for Container in self.barge.load:
                    if Container.origin != 0:
                        if Container.destination == self.terminal.id or
Container.destination == 7:
                            Container.leave(self.barge.load)
                            Container.enter(self.terminal.stack)
                            self.terminal.unloaded_from_barge.trigger()
                    elif Container.origin == 0 and Container.destination
== self.terminal.id:
                        Container.leave(self.barge.load)
                        # self.myterminal.delivered += 1
                        if Container.due_time < env._now:
                            env.non_performance_count += 1
                            monitor_non_performance_rate.tally(
                                env.non_performance_count * 100 /
env.containers_generated)
                            self.terminal.late_delivery += 1
                            self.terminal.late_import += 1
                            print("{0:.2f}".format(env._now)+' non-
performance registered by BC, import containers late arrival: ' +
"{0:.2f}".format(
                                    env.non_performance_count * 100 /
env.containers_generated) + ' %')
                        else:
                            self.terminal.ontime_delivery += 1
            else:
            ########################################################
            ##activate for case 1 and 2 and 3##
                for Container in self.barge.load:
                    if Container.origin != 0 and Container.destination ==
self.terminal.id:
                        Container.leave(self.barge.load)
                        Container.enter(self.terminal.stack)
                        self.terminal.unloaded_from_barge.trigger()
                    elif Container.origin == 0 and Container.destination
== self.terminal.id:
                        Container.leave(self.barge.load)
                        # self.myterminal.delivered += 1
                        if Container.due_time < env._now:
                            # print('BC register non-performance while
unloading barge')
```

```
                            env.non_performance_count += 1
                            monitor_non_performance_rate.tally(
                                env.non_performance_count * 100 /
env.containers_generated)
                            self.terminal.late_delivery += 1
                            self.terminal.late_import += 1
                            print("{0:.2f}".format(env._now)+' non-
performance registered by BC, import containers late arrival: ' +
"{0:.2f}".format(
                                        env.non_performance_count * 100 /
env.containers_generated) + ' %')
                        else:
                            self.terminal.ontime_delivery += 1
            ##########################################################

            # load

            ##activate for case 3 only##
            if self.barge.name() =='PBS2' and self.terminal == Terminal3:
                for Container in self.terminal.stack:
                    if Container.destination == 7:
                        self.container = Container
                        self.terminal.stack.remove(self.container)
                        self.container.enter(self.barge.load)
                        break
                else:
                    yield self.wait(self.terminal.unloaded_from_vessel,
incoming_from_hinterland)
            else:
            ##########################################################
            # activate for case 1 and 2 and 3##
                for Container in self.terminal.stack:
                    if Container.destination != self.terminal.id:
                        # self.container = Container
                        Container.leave(self.terminal.stack)
                        Container.enter(self.barge.load)
                        break
                else:
                    yield self.wait(self.terminal.unloaded_from_vessel,
incoming_from_hinterland)
            ##########################################################


################                                  ##########################
################ SIMULATION INITIALIZATION ##########################
################                                  ##########################

env = sim.Environment(trace=False, time_unit='minutes')
env.background_color('20%gray')
env.animate(False)
env.speed(env.minutes(8))
env.modelname('Space@Sea Simulation')

Weather = Weather()
env.non_performance_count = 0
env.containers_generated = 0

incoming_from_hinterland = sim.State('incoming from hinterland')
```

```
monitor_non_performance_rate = sim.Monitor(name='Non-performance rate',
level=True, initial_tally=0)

## create terminals ##

# for port input specification
terminal_index = {'Terminal1': 1, 'Terminal2': 2, 'Terminal3': 3,
                  'Terminal4': 4, 'Terminal5': 5, 'Terminal6': 6,
                  'SpaceAtSeaPlatform': 7}

Terminal1 = Terminal(name='Terminal1')
Terminal2 = Terminal(name='Terminal2')
Terminal3 = Terminal(name='Terminal3')
Terminal4 = Terminal(name='Terminal4')
Terminal5 = Terminal(name='Terminal5')
Terminal6 = Terminal(name='Terminal6')
SpaceAtSeaPlatform = Terminal(name='SpaceAtSeaPlatform')

# define dictionaries for the origin of the Containers

container_origin = {1: Terminal1.stack, 2: Terminal2.stack, 3:
Terminal3.stack,
                    4: Terminal4.stack, 5: Terminal5.stack, 6:
Terminal6.stack}

# define dictionaries for the destination of the Vessel

vessel_destination = {1: Terminal1, 2: Terminal2, 3: Terminal3, 4:
Terminal4,
                      5: Terminal5, 6: Terminal6, 7: SpaceAtSeaPlatform}

# define dictionaries for the PBS routes

PBS_route = {'PBS1': 'case1.txt',
             'PBS2': 'case3.txt',
             'PBS3': 'case1-3.txt'}

## create PBS  ##

PBS1 = Barge(name='PBS1')
PBS2 = Barge(name='PBS2')
# PBS3 = Barge(name='PBS3')

## create vessel generator ##

VesselGenerator(name='VesselGenerator')

## animation  ##

# non-perfomance rate

sim.AnimateText(text=lambda: 'Non-performance rate: '+"{0:.2f}".format(
    env.non_performance_count * 100 / env.containers_generated)+' %',
x=500, y=590, fontsize=16)

sim.AnimateText(text=lambda: 'Number of containers generated:
'+"{0:.0f}".format(
                  env.containers_generated), x=500, y=580, fontsize=12)
```

```
sim.AnimateMonitor(monitor_non_performance_rate, x=500, y=520, width=200,
height=50,
                   horizontal_scale=1/500, vertical_scale=1/2, title='')

# PBS status and graphs

sim.AnimateText(text='PBS status:', x=450, y=430, fontsize=20)
# PBS1
sim.AnimateMonitor(PBS1.load.length, x=440, y=380, width=200, height=25,
                   horizontal_scale=1/500, vertical_scale=1/100,
                   title=lambda: '#1 Location: ' + str(PBS1.terminal) +
'\n     Load: ' + str(len(PBS1.load)) + ' TEUs',
                   titlefontsize=12)
# PBS2
sim.AnimateMonitor(PBS2.load.length, x=440, y=325, width=200, height=25,
horizontal_scale=1/500,
                   vertical_scale=1/100,
                   title=lambda: '#2 Location: ' + str(PBS2.terminal) +
'\n     Load: ' + str(len(PBS2.load)) + ' TEUs',
                   titlefontsize=12)
# # PBS3
# sim.AnimateMonitor(PBS3.load.length, x=440, y=270, width=200, height=25,
horizontal_scale=1/500,
#                    vertical_scale=1/100,
#                    title=lambda: '#3 Location: ' + str(PBS3.terminal) +
'\n     Load: ' + str(len(PBS3.load)) + ' TEUs',
#                    titlefontsize=12)

# # weather

sim.AnimateText(text='Weather factor at platform:', x=375, y=640,
fontsize=14)
sim.AnimateText(text=lambda: 'Wind: ' +
"{0:.2f}".format(Weather.wind_efficiency_at_platform),
                x=385, y=625, fontsize=12)
sim.AnimateText(text=lambda: 'Sea wave: ' +
"{0:.2f}".format(Weather.wave_efficiency_at_platform),
                x=385, y=610, fontsize=12)

# map and terminal indicator lines
sim.AnimateLine(spec=(390, 285, 450, 250), linecolor='cyan', linewidth=4,
layer=1)
sim.AnimateLine(spec=(450, 250, 750, 200), linecolor='cyan', linewidth=4,
layer=1)
sim.AnimateLine(spec=(750, 200, 650, 420), linecolor='cyan', linewidth=4,
layer=1)
sim.AnimateLine(spec=(650, 420, 400, 570), linecolor='cyan', linewidth=4,
layer=1)
sim.AnimateLine(spec=(450, 500, 400, 570), linecolor='cyan', linewidth=4,
layer=1)
sim.AnimateLine(spec=(390, 285, 450, 500), linecolor='cyan', linewidth=4,
layer=1)

# terminal1 indicator

sim.AnimateLine(spec=(400, 570, 330, 640))
sim.AnimateCircle(radius=10, x=400, y=570, layer=0) # terminal1
qa43 = sim.AnimateQueue(Terminal1.bargeQ, title='', x=400, y=570)
```

```
sim.AnimateRectangle(spec=(40, 480, 330, 640), fillcolor='',
linecolor='white')
sim.AnimateText(text='Terminal 1', x=50, y=640, fontsize=20)

# terminal2 indicator

sim.AnimateLine(spec=(450, 500, 340, 440))
sim.AnimateCircle(radius=10, x=450, y=500, layer=0) # terminal2
qa44 = sim.AnimateQueue(Terminal2.bargeQ, title='', x=450, y=500)
sim.AnimateRectangle(spec=(50, 280, 340,
440),fillcolor='',linecolor='white')
sim.AnimateText(text='Terminal 2', x=60, y=440, fontsize=20)

# terminal3 indicator

sim.AnimateLine(spec=(330, 210, 390, 285))
sim.AnimateCircle(radius=10, x=390, y=285, layer=0) # terminal3
qa45 = sim.AnimateQueue(Terminal3.bargeQ, title='', x=390, y=285)
sim.AnimateRectangle(spec=(40, 50, 330, 210), fillcolor='',
linecolor='white')
sim.AnimateText(text='Terminal 3', x=50, y=210, fontsize=20)

# terminal4 indicator

sim.AnimateLine(spec=(450, 250, 370, 180))
sim.AnimateCircle(radius=10, x=450, y=250, layer=0) # terminal4
qa46 = sim.AnimateQueue(Terminal4.bargeQ, title='', x=450, y=250)
sim.AnimateRectangle(spec=(370, 20, 660, 180), fillcolor='',
linecolor='white')
sim.AnimateText(text='Terminal 4', x=560, y=180, fontsize=20)

# terminal5 indicator

sim.AnimateLine(spec=(720, 170, 750, 200))
sim.AnimateCircle(radius=10, x=750, y=200, layer=0) # terminal5
qa47 = sim.AnimateQueue(Terminal5.bargeQ, title='', x=750, y=200)
sim.AnimateRectangle(spec=(720, 10, 1010, 170), fillcolor='',
linecolor='white')
sim.AnimateText(text='Terminal 5', x=910, y=170, fontsize=20)

# terminal6 indicator

sim.AnimateLine(spec=(720, 460, 650, 420))
sim.AnimateCircle(radius=10, x=650, y=420, layer=0)
qa48 = sim.AnimateQueue(Terminal6.bargeQ, title='', x=650, y=420)
sim.AnimateRectangle(spec=(720, 300, 1010, 460), fillcolor='',
linecolor='white')
sim.AnimateText(text='Terminal 6', x=910, y=460, fontsize=20)

# space@sea indicator

sim.AnimateLine(spec=(400, 680, 720, 710))
sim.AnimateCircle(radius=10, x=400, y=680, layer=0) #space@sea
qa49 = sim.AnimateQueue(SpaceAtSeaPlatform.bargeQ, title='', x=400, y=680)
sim.AnimateRectangle(spec=(720, 550, 1010, 710), fillcolor='',
linecolor='white')
sim.AnimateText(text='Space@Sea Platform', x=830, y=710, fontsize=20)
#
#
```

```
# terminal 1 graph
sim.AnimateMonitor(Terminal1.vesselQ.length, x=50, y=590, width=200,
height=25,
                   horizontal_scale=1/500, vertical_scale=2,
                   title=lambda:'Vessel queue: ' +
"{0:.0f}".format(len(Terminal1.vesselQ)))

sim.AnimateMonitor(Terminal1.stack.length, x=50, y=540, width=200,
height=25,
                   horizontal_scale=1/500, vertical_scale=1/3000,
                   title=lambda:'Stack: ' +
"{0:.0f}".format(len(Terminal1.stack)) + ' TEUs')

sim.AnimateText(text=lambda: 'Mean dwell time: ' + "{0:.2f}".format(
    Terminal1.stack.length_of_stay.mean() / 1440) + ' days', x=180, y=568,
fontsize=12)

sim.AnimateText(text=lambda: 'On-time:\n' +
"{0:.0f}".format(Terminal1.ontime_delivery), x=255, y=540,
                fontsize=12)

sim.AnimateText(text=lambda: 'Late (import):\n' + "{0:.0f}".format(
    Terminal1.late_delivery) + " (" +
"{0:.0f}".format(Terminal1.late_import) + ")", x=255, y=515, fontsize=12)

sim.AnimateMonitor(Terminal1.quaycraneQ.length, x=50, y=490, width=200,
height=25,
                   horizontal_scale=1/500, vertical_scale=1.5,
                   title=lambda:'Quay Crane: ' + "{0:.2f}".format(
        (Terminal1.noofquaycrane - len(Terminal1.quaycraneQ)) * 100 /
Terminal1.noofquaycrane) + ' %')


# terminal 2 graph
sim.AnimateMonitor(Terminal2.vesselQ.length, x=60, y=390, width=200,
height=25,
                   horizontal_scale=1/500, vertical_scale=2,
                   title=lambda:'Vessel queue: ' +
"{0:.0f}".format(len(Terminal2.vesselQ)))

sim.AnimateMonitor(Terminal2.stack.length, x=60, y=340, width=200,
height=25,
                   horizontal_scale=1/500, vertical_scale=1/3000,
                   title=lambda:'Stack: ' +
"{0:.0f}".format(len(Terminal2.stack)) + ' TEUs')

sim.AnimateText(text=lambda: 'Mean dwell time: ' + "{0:.2f}".format(
    Terminal2.stack.length_of_stay.mean() / 1440) + ' days', x=190, y=368,
fontsize=12)

sim.AnimateText(text=lambda: 'On-time:\n' +
"{0:.0f}".format(Terminal2.ontime_delivery), x=265, y=340,
                fontsize=12)

sim.AnimateText(text=lambda: 'Late (import):\n' + "{0:.0f}".format(
    Terminal1.late_delivery) + " (" +
"{0:.0f}".format(Terminal2.late_import) + ")", x=265, y=315, fontsize=12)
```

```
sim.AnimateMonitor(Terminal2.quaycraneQ.length, x=60, y=290, width=200,
height=25,
                   horizontal_scale=1/500, vertical_scale=2.4,
                   title=lambda:'Quay Crane: ' + "{0:.2f}".format(
        (Terminal2.noofquaycrane - len(Terminal2.quaycraneQ)) * 100 /
Terminal2.noofquaycrane) + ' %')


# terminal 3 graph
sim.AnimateMonitor(Terminal3.vesselQ.length, x=50, y=160, width=200,
height=25,
                   horizontal_scale=1/500, vertical_scale=2,
                   title=lambda:'Vessel queue: ' +
"{0:.0f}".format(len(Terminal3.vesselQ)))

sim.AnimateMonitor(Terminal3.stack.length, x=50, y=110, width=200,
height=25,
                   horizontal_scale=1/500, vertical_scale=1/3000,
                   title=lambda:'Stack: ' +
"{0:.0f}".format(len(Terminal3.stack)) + ' TEUs')

sim.AnimateText(text=lambda: 'Mean dwell time: ' + "{0:.2f}".format(
    Terminal3.stack.length_of_stay.mean() / 1440) + ' days', x=180, y=138,
fontsize=12)

sim.AnimateText(text=lambda: 'On-time:\n' +
"{0:.0f}".format(Terminal3.ontime_delivery), x=255, y=110,
                fontsize=12)

sim.AnimateText(text=lambda: 'Late (import):\n' + "{0:.0f}".format(
    Terminal1.late_delivery) + " (" +
"{0:.0f}".format(Terminal3.late_import) + ")", x=255, y=85, fontsize=12)

sim.AnimateMonitor(Terminal3.quaycraneQ.length, x=50, y=60, width=200,
height=25,
                   horizontal_scale=1/500, vertical_scale=0.45,
                   title=lambda:'Quay Crane: ' + "{0:.2f}".format(
        (Terminal3.noofquaycrane - len(Terminal3.quaycraneQ)) * 100 /
Terminal3.noofquaycrane) + ' %')


# terminal 4 graph
sim.AnimateMonitor(Terminal4.vesselQ.length, x=380, y=130, width=200,
height=25,
                   horizontal_scale=1/500, vertical_scale=2,
                   title=lambda:'Vessel queue: ' +
"{0:.0f}".format(len(Terminal4.vesselQ)))

sim.AnimateMonitor(Terminal4.stack.length, x=380, y=80, width=200,
height=25,
                   horizontal_scale=1/500, vertical_scale=1/3000,
                   title=lambda:'Stack: ' +
"{0:.0f}".format(len(Terminal4.stack)) + ' TEUs')

sim.AnimateText(text=lambda: 'Mean dwell time: ' + "{0:.2f}".format(
    Terminal4.stack.length_of_stay.mean() / 1440) + ' days', x=510, y=108,
fontsize=12)
```

```
sim.AnimateText(text=lambda: 'On-time:\n' +
"{0:.0f}".format(Terminal4.ontime_delivery), x=585, y=80,
                fontsize=12)

sim.AnimateText(text=lambda: 'Late (import):\n' + "{0:.0f}".format(
    Terminal1.late_delivery) + " (" +
"{0:.0f}".format(Terminal4.late_import) + ")", x=585, y=55, fontsize=12)

sim.AnimateMonitor(Terminal4.quaycraneQ.length, x=380, y=30, width=200,
height=25,
                   horizontal_scale=1/500, vertical_scale=1.5,
                   title=lambda:'Quay Crane: ' + "{0:.2f}".format(
        (Terminal4.noofquaycrane - len(Terminal4.quaycraneQ)) * 100 /
Terminal4.noofquaycrane) + ' %')



# terminal 5 graph
sim.AnimateMonitor(Terminal5.vesselQ.length, x=730, y=120, width=200,
height=25,
                   horizontal_scale=1/500, vertical_scale=2,
                   title=lambda:'Vessel queue: ' +
"{0:.0f}".format(len(Terminal5.vesselQ)))

sim.AnimateMonitor(Terminal5.stack.length, x=730, y=70, width=200,
height=25,
                   horizontal_scale=1/500, vertical_scale=1/3000,
                   title=lambda:'Stack: ' +
"{0:.0f}".format(len(Terminal5.stack)) + ' TEUs')

sim.AnimateText(text=lambda: 'Mean dwell time: ' + "{0:.2f}".format(
    Terminal5.stack.length_of_stay.mean() / 1440) + ' days', x=860, y=98,
fontsize=12)

sim.AnimateText(text=lambda: 'On-time:\n' +
"{0:.0f}".format(Terminal5.ontime_delivery), x=935, y=70,
                fontsize=12)

sim.AnimateText(text=lambda: 'Late (import):\n' + "{0:.0f}".format(
    Terminal1.late_delivery) + " (" +
"{0:.0f}".format(Terminal5.late_import) + ")", x=935, y=45, fontsize=12)

sim.AnimateMonitor(Terminal5.quaycraneQ.length, x=730, y=20, width=200,
height=25,
                   horizontal_scale=1/500, vertical_scale=1.5,
                   title=lambda:'Quay Crane: ' + "{0:.2f}".format(
        (Terminal5.noofquaycrane - len(Terminal5.quaycraneQ)) * 100 /
Terminal5.noofquaycrane) + ' %')


# terminal 6 graph
sim.AnimateMonitor(Terminal6.vesselQ.length, x=730, y=410, width=200,
height=25,
                   horizontal_scale=1/500, vertical_scale=2,
                   title=lambda:'Vessel queue: ' +
"{0:.0f}".format(len(Terminal6.vesselQ)))

sim.AnimateMonitor(Terminal6.stack.length, x=730, y=360, width=200,
height=25,
```

```
                    horizontal_scale=1/500, vertical_scale=1/3000,
                    title=lambda:'Stack: ' +
"{0:.0f}".format(len(Terminal6.stack)) + ' TEUs')

sim.AnimateText(text=lambda: 'Mean dwell time: ' + "{0:.2f}".format(
    Terminal6.stack.length_of_stay.mean() / 1440) + ' days', x=860, y=388,
fontsize=12)

sim.AnimateText(text=lambda: 'On-time:\n' +
"{0:.0f}".format(Terminal6.ontime_delivery), x=935, y=360,
                fontsize=12)

sim.AnimateText(text=lambda: 'Late (import):\n' + "{0:.0f}".format(
    Terminal1.late_delivery) + " (" +
"{0:.0f}".format(Terminal6.late_import) + ")", x=935, y=335, fontsize=12)

sim.AnimateMonitor(Terminal6.quaycraneQ.length, x=730, y=310, width=200,
height=25,
                    horizontal_scale=1/500, vertical_scale=1.5,
                    title=lambda:'Quay Crane: ' + "{0:.2f}".format(
        (Terminal6.noofquaycrane - len(Terminal6.quaycraneQ)) * 100 /
Terminal6.noofquaycrane) + ' %')


# space at sea graph
sim.AnimateMonitor(SpaceAtSeaPlatform.vesselQ.length, x=730, y=660,
width=200, height=25,
                    horizontal_scale=1/500, vertical_scale=2,
                    title=lambda:'Vessel queue: ' +
"{0:.0f}".format(len(SpaceAtSeaPlatform.vesselQ)))

sim.AnimateMonitor(SpaceAtSeaPlatform.stack.length, x=730, y=610,
width=200, height=25,
                    horizontal_scale=1/500, vertical_scale=1/3000,
                    title=lambda:'Stack: ' +
"{0:.0f}".format(len(SpaceAtSeaPlatform.stack)) + ' TEUs')

sim.AnimateText(text=lambda: 'Mean dwell time: ' + "{0:.2f}".format(
    SpaceAtSeaPlatform.stack.length_of_stay.mean() / 1440) + ' days',
x=860, y=638, fontsize=12)

sim.AnimateText(text=lambda: 'On-time:\n' +
"{0:.0f}".format(SpaceAtSeaPlatform.ontime_delivery), x=935, y=610,
                fontsize=12)

sim.AnimateText(text=lambda: 'Late (import):\n' + "{0:.0f}".format(
    SpaceAtSeaPlatform.late_delivery) + " (" +
"{0:.0f}".format(SpaceAtSeaPlatform.late_import) + ")", x=935, y=585,
                fontsize=12)

sim.AnimateMonitor(SpaceAtSeaPlatform.quaycraneQ.length, x=730, y=560,
width=200, height=25,
                    horizontal_scale=1/500, vertical_scale=3,
                    title=lambda:'Quay Crane: ' +
"{0:.2f}".format((SpaceAtSeaPlatform.noofquaycrane - len(
                        SpaceAtSeaPlatform.quaycraneQ)) * 100 /
SpaceAtSeaPlatform.noofquaycrane) + ' %')
```

```
sim.AnimateText(text=lambda: 'Booked capacity: ' +
"{0:.0f}".format(SpaceAtSeaPlatform.booked_capacity), x=860, y=690,
                fontsize=14)

# # terminal 1 stats
#
# sim.AnimateText(text=lambda: 'Vessel queue: ' +
"{0:.0f}".format(len(Terminal1.vesselQ)), x=230, y=620, fontsize=14)
# #qa00 = sim.AnimateQueue(Terminal1.VesselQ, title='', x=260, y=605,
direction='s', max_length=9)
# sim.AnimateText(text=lambda: 'Stack length: ' +
"{0:.0f}".format(len(Terminal1.stack)), x=50, y=620, fontsize=14)
# sim.AnimateText(text=lambda: 'Mean dwell time: ' + "{0:.4f}".format(
#     Terminal1.stack.length_of_stay.mean() / 1440) + ' days', x=55,
y=605, fontsize=12)
# sim.AnimateText(text=lambda: 'On-time deliveries: ' +
"{0:.0f}".format(Terminal1.ontime_delivery), x=55, y=590,
#                 fontsize=12)
# sim.AnimateText(text=lambda: 'Late deliveries (import): ' +
"{0:.0f}".format(
#     Terminal1.late_delivery) + " (" +
"{0:.0f}".format(Terminal1.late_import) + ")", x=55, y=575, fontsize=12)
# sim.AnimateText(text=lambda: 'Quay Cranes: ' + "{0:.2f}".format(
#     (Terminal1.noofquaycrane - len(Terminal1.quaycraneQ)) * 100 /
Terminal1.noofquaycrane) + ' %', x=50, y=555,
#                 fontsize=14)
# # sim.AnimateText(text=lambda:'Booked QC:
'+"{0:.2f}".format(Terminal1.bookedQC), x=50, y=540, fontsize=12)
# sim.AnimateText(text=lambda:'Upcoming call: ' +
"{0:.0f}".format(Terminal1.upcoming_call), x=50, y=525, fontsize=14)
# #qa03 = sim.AnimateQueue(Terminal1.QuayCraneQ, title='', x=55, y=550,
direction='e')
# #sim.AnimateText(text='Barge Crane:', x=50, y=495, fontsize=14)
# #qa05 = sim.AnimateQueue(Terminal1.BargeCraneQ, title='', x=55, y=490,
direction='e')
#
# # terminal 2 stats
#
# sim.AnimateText(text=lambda: 'Vessel queue: ' +
"{0:.0f}".format(len(Terminal2.vesselQ)), x=240, y=420, fontsize=14)
# #qa06 = sim.AnimateQueue(Terminal2.VesselQ, title='', x=270, y=405,
direction='s', max_length=9)
# sim.AnimateText(text=lambda: 'Stack length: ' +
"{0:.0f}".format(len(Terminal2.stack)), x=60, y=420, fontsize=14)
# sim.AnimateText(text=lambda: 'Mean dwell time: ' + "{0:.4f}".format(
#     Terminal2.stack.length_of_stay.mean() / 1440) + ' days', x=65,
y=405, fontsize=12)
# sim.AnimateText(text=lambda: 'On-time deliveries: ' +
"{0:.0f}".format(Terminal2.ontime_delivery), x=65, y=390,
#                 fontsize=12)
# sim.AnimateText(text=lambda: 'Late deliveries (import): ' +
"{0:.0f}".format(
#     Terminal2.late_delivery) + " (" +
"{0:.0f}".format(Terminal2.late_import) + ")", x=65, y=375, fontsize=12)
# sim.AnimateText(text=lambda: 'Quay Cranes: ' + "{0:.2f}".format(
#     (Terminal2.noofquaycrane - len(Terminal2.quaycraneQ)) * 100 /
Terminal2.noofquaycrane) + ' %', x=60, y=355,
#                 fontsize=14)
```

```
# # sim.AnimateText(text=lambda:'Booked QC:
'+"{0:.2f}".format(Terminal2.bookedQC), x=60, y=340, fontsize=12)
# sim.AnimateText(text=lambda: 'Upcoming call: ' +
"{0:.0f}".format(Terminal2.upcoming_call), x=60, y=325, fontsize=14)
# # qa09 = sim.AnimateQueue(Terminal2.QuayCraneQ, title='', x=65, y=350,
direction='e')
# # sim.AnimateText(text='Barge Crane:', x=60, y=295, fontsize=14)
# # qa11 = sim.AnimateQueue(Terminal2.BargeCraneQ, title='', x=65, y=290,
direction='e')
#
# # terminal 3 stats
#
# sim.AnimateText(text=lambda: 'Vessel queue: ' +
"{0:.0f}".format(len(Terminal3.vesselQ)), x=230, y=190, fontsize=14)
# #qa12 = sim.AnimateQueue(Terminal3.VesselQ, title='', x=260, y=175,
direction='s', max_length=9)
# sim.AnimateText(text=lambda: 'Stack length: ' +
"{0:.0f}".format(len(Terminal3.stack)), x=50, y=190, fontsize=14)
# #sim.AnimateText(text=lambda:str(len(Terminal3.export_stack)),x=125,
y=190, fontsize=14)
# sim.AnimateText(text=lambda: 'Mean dwell time: ' + "{0:.4f}".format(
#     Terminal3.stack.length_of_stay.mean() / 1440) + ' days', x=55,
y=175, fontsize=12)
# sim.AnimateText(text=lambda: 'On-time deliveries: ' +
"{0:.0f}".format(Terminal3.ontime_delivery), x=55, y=160,
#                 fontsize=12)
# sim.AnimateText(text=lambda: 'Late deliveries (import): ' +
"{0:.0f}".format(
#     Terminal3.late_delivery) + " (" +
"{0:.0f}".format(Terminal3.late_import)+")", x=55, y=145, fontsize=12)
# sim.AnimateText(text=lambda:'Quay Cranes: '+"{0:.2f}".format(
#     (Terminal3.noofquaycrane-
len(Terminal3.quaycraneQ))*100/Terminal3.noofquaycrane)+' %', x=50, y=125,
fontsize=14)
# # sim.AnimateText(text=lambda:'Booked QC:
'+"{0:.2f}".format(Terminal3.bookedQC), x=50, y=110, fontsize=12)
# sim.AnimateText(text=lambda: 'Upcoming call:
'+"{0:.0f}".format(Terminal3.upcoming_call), x=50, y=95, fontsize=14)
# # qa15 = sim.AnimateQueue(Terminal3.QuayCraneQ, title='', x=55, y=120,
direction='e')
# # sim.AnimateText(text='Barge Crane:', x=50, y=65, fontsize=14)
# # qa17 = sim.AnimateQueue(Terminal3.BargeCraneQ, title='', x=55, y=60,
direction='e')

# # terminal 4 stats

# sim.AnimateText(text=lambda: 'Vessel queue: ' +
"{0:.0f}".format(len(Terminal4.vesselQ)), x=385, y=160, fontsize=14)
# #qa18 = sim.AnimateQueue(Terminal4.VesselQ, title='', x=415, y=145,
direction='s', max_length=9)
# sim.AnimateText(text=lambda: 'Stack length: ' +
"{0:.0f}".format(len(Terminal4.stack)), x=500, y=160, fontsize=14)
# #sim.AnimateText(text=lambda:str(len(Terminal4.export_stack)),x=575,
y=160, fontsize=14)
# sim.AnimateText(text=lambda: 'Mean dwell time: ' + "{0:.4f}".format(
#     Terminal4.stack.length_of_stay.mean() / 1440) + ' days', x=505,
y=145, fontsize=12)
# sim.AnimateText(text=lambda: 'On-time deliveries: ' +
"{0:.0f}".format(Terminal4.ontime_delivery), x=505, y=130,
```

```
#                 fontsize=12)
# sim.AnimateText(text=lambda: 'Late deliveries (import): ' +
"{0:.0f}".format(
#     Terminal4.late_delivery) + " (" +
"{0:.0f}".format(Terminal4.late_import) + ")", x=505, y=115, fontsize=12)
# sim.AnimateText(text=lambda: 'Quay Cranes: ' + "{0:.2f}".format(
#     (Terminal4.noofquaycrane - len(Terminal4.quaycraneQ)) * 100 /
Terminal4.noofquaycrane) + ' %', x=500, y=95,
#                 fontsize=14)
# # sim.AnimateText(text=lambda:'Booked QC:
'+"{0:.2f}".format(Terminal3.bookedQC), x=50, y=110, fontsize=12)
# sim.AnimateText(text=lambda: 'Upcoming call: ' +
"{0:.0f}".format(Terminal4.upcoming_call), x=500, y=65, fontsize=14)
# # qa21 = sim.AnimateQueue(Terminal4.QuayCraneQ, title='', x=505, y=90,
direction='e')
# # sim.AnimateText(text='Barge Crane:', x=500, y=35, fontsize=14)
# # qa23 = sim.AnimateQueue(Terminal4.BargeCraneQ, title='', x=505, y=30,
direction='e')

# # terminal 5 stats

# sim.AnimateText(text=lambda: 'Vessel queue: ' +
"{0:.0f}".format(len(Terminal5.vesselQ)), x=735, y=150, fontsize=14)
# #qa24 = sim.AnimateQueue(Terminal5.VesselQ, title='', x=765, y=135,
direction='s', max_length=9)
# sim.AnimateText(text=lambda: 'Stack length: ' +
"{0:.0f}".format(len(Terminal5.stack)), x=850, y=150, fontsize=14)
# sim.AnimateText(text=lambda: 'Mean dwell time: ' + "{0:.4f}".format(
#     Terminal5.stack.length_of_stay.mean() / 1440) + ' days', x=855,
y=135, fontsize=12)
# sim.AnimateText(text=lambda: 'On-time deliveries: ' +
"{0:.0f}".format(Terminal5.ontime_delivery), x=855, y=120,
#                 fontsize=12)
# sim.AnimateText(text=lambda: 'Late deliveries (import): ' +
"{0:.0f}".format(
#     Terminal5.late_delivery) + " (" +
"{0:.0f}".format(Terminal5.late_import) + ")", x=855, y=105, fontsize=12)
# sim.AnimateText(text=lambda: 'Quay Cranes: ' + "{0:.2f}".format(
#     (Terminal5.noofquaycrane - len(Terminal5.quaycraneQ)) * 100 /
Terminal5.noofquaycrane) + ' %', x=850, y=85,
#                 fontsize=14)
# # sim.AnimateText(text=lambda:'Booked QC:
'+"{0:.2f}".format(Terminal3.bookedQC), x=50, y=110, fontsize=12)
# sim.AnimateText(text=lambda: 'Upcoming call: ' +
"{0:.0f}".format(Terminal5.upcoming_call), x=850, y=55, fontsize=14)
# # qa27 = sim.AnimateQueue(Terminal5.QuayCraneQ, title='', x=855, y=80,
direction='e')
# # sim.AnimateText(text='Barge Crane:', x=850, y=25, fontsize=14)
# # qa29 = sim.AnimateQueue(Terminal5.BargeCraneQ, title='', x=855, y=20,
direction='e')

# # terminal 6 stats

# sim.AnimateText(text=lambda: 'Vessel queue: ' +
"{0:.0f}".format(len(Terminal6.vesselQ)), x=735, y=440, fontsize=14)
# #qa30 = sim.AnimateQueue(Terminal6.VesselQ, title='', x=765, y=425,
direction='s', max_length=9)
# sim.AnimateText(text=lambda: 'Stack length: ' +
"{0:.0f}".format(len(Terminal6.stack)), x=850, y=440, fontsize=14)
```

```
# sim.AnimateText(text=lambda: 'Mean dwell time: ' + "{0:.4f}".format(
#     Terminal6.stack.length_of_stay.mean() / 1440) + ' days', x=855,
y=425, fontsize=12)
# sim.AnimateText(text=lambda: 'On-time deliveries: ' +
"{0:.0f}".format(Terminal6.ontime_delivery), x=855, y=410,
#                 fontsize=12)
# sim.AnimateText(text=lambda: 'Late deliveries (import): ' +
"{0:.0f}".format(
#     Terminal6.late_delivery) + " (" +
"{0:.0f}".format(Terminal6.late_import) + ")", x=855, y=395, fontsize=12)
# sim.AnimateText(text=lambda: 'Quay Cranes: ' + "{0:.2f}".format(
#     (Terminal6.noofquaycrane - len(Terminal6.quaycraneQ)) * 100 /
Terminal6.noofquaycrane) + ' %', x=850, y=375,
#                 fontsize=14)
# # sim.AnimateText(text=lambda:'Booked QC:
'+"{0:.2f}".format(Terminal3.bookedQC), x=50, y=110, fontsize=12)
# sim.AnimateText(text=lambda: 'Upcoming call: ' +
"{0:.0f}".format(Terminal6.upcoming_call), x=850, y=345, fontsize=14)
# # qa33 = sim.AnimateQueue(Terminal6.QuayCraneQ, title='', x=855, y=370,
direction='e')
# # sim.AnimateText(text='Barge Crane:', x=850, y=315, fontsize=14)
# # qa35 = sim.AnimateQueue(Terminal6.BargeCraneQ, title='', x=855, y=310,
direction='e')

# # space@sea stats

# sim.AnimateText(text=lambda: 'Vessel queue: ' +
"{0:.0f}".format(len(SpaceAtSeaPlatform.vesselQ)), x=735, y=690,
#                 fontsize=14)
# #qa36 = sim.AnimateQueue(SpaceAtSeaPlatform.VesselQ, title='', x=765,
y=675, direction='s', max_length=9)
# sim.AnimateText(text=lambda: 'Stack length: ' +
"{0:.0f}".format(len(SpaceAtSeaPlatform.stack)), x=850, y=690,
#                 fontsize=14)
# sim.AnimateText(text=lambda: 'Mean dwell time: ' + "{0:.4f}".format(
#     SpaceAtSeaPlatform.stack.length_of_stay.mean() / 1440) + ' days',
x=855, y=675, fontsize=12)
# sim.AnimateText(text=lambda: 'On-time deliveries: ' +
"{0:.0f}".format(SpaceAtSeaPlatform.ontime_delivery), x=855,
#                 y=660, fontsize=12)
# sim.AnimateText(text=lambda: 'Late deliveries: ' + "{0:.0f}".format(
#     SpaceAtSeaPlatform.late_delivery) + " (" +
"{0:.0f}".format(SpaceAtSeaPlatform.late_import) + ")", x=855, y=645,
#                 fontsize=12)
# sim.AnimateText(text=lambda: 'Quay Cranes: ' + "{0:.2f}".format(
#     (SpaceAtSeaPlatform.noofquaycrane -
len(SpaceAtSeaPlatform.quaycraneQ))
#     * 100 / SpaceAtSeaPlatform.noofquaycrane) + ' %', x=850, y=625,
fontsize=14)
# # sim.AnimateText(text=lambda:'Booked QC:
'+"{0:.2f}".format(SpaceAtSeaPlatform.bookedQC), x=850, y=610,
fontsize=12)
# sim.AnimateText(text=lambda: 'Upcoming call: ' +
"{0:.0f}".format(SpaceAtSeaPlatform.upcoming_call), x=850, y=595,
#                 fontsize=14)
# sim.AnimateText(text=lambda: 'Booked capacity: ' +
"{0:.0f}".format(SpaceAtSeaPlatform.booked_capacity), x=850, y=565,
#                 fontsize=14)
```

```
# # qa39 = sim.AnimateQueue(SpaceAtSeaPlatform.QuayCraneQ, title='',
x=855, y=620, direction='e')
# # sim.AnimateText(text='Barge Crane:', x=850, y=565, fontsize=14)
# # qa41 = sim.AnimateQueue(SpaceAtSeaPlatform.BargeCraneQ, title='',
x=855, y=560, direction='e')

# # container queue animation
# qa02 = sim.AnimateQueue(Terminal1.stack, title='', x=55, y=594,
direction='e', max_length=15, reverse=True)
# qa08 = sim.AnimateQueue(Terminal2.stack, title='', x=65, y=394,
direction='e', max_length=15, reverse=True)
# qa14 = sim.AnimateQueue(Terminal3.stack, title='', x=55, y=164,
direction='e', max_length=15, reverse=True)
# qa20 = sim.AnimateQueue(Terminal4.stack, title='', x=643, y=134,
direction='w', max_length=15, reverse=True)
# qa26 = sim.AnimateQueue(Terminal5.stack, title='', x=993, y=124,
direction='w', max_length=15, reverse=True)
# qa32 = sim.AnimateQueue(Terminal6.stack, title='', x=993, y=414,
direction='w', max_length=15, reverse=True)
# qa38 = sim.AnimateQueue(SpaceAtSeaPlatform.stack, title='', x=993,
y=664, direction='w', max_length=15, reverse=True)
# qa42 = sim.AnimateQueue(PBS1.myload, title='', x=450, y=320,
direction='e', reverse=True, max_length=15)

# # non-performance graph only

sim.AnimateText(text=lambda: 'PBS status:\n'
                             '1 Location: ' + str(PBS1.terminal)+'\n'
                             '  Load: ' + str(len(PBS1.load)) + ' TEUs\n',
                             # '2 Location: ' + str(PBS2.terminal)+'\n'
                             # '  Load: ' + str(len(PBS2.load)) + '
TEUs\n'
                             # '3 Location: ' + str(PBS3.terminal)+'\n'
                             # '  Load: ' + str(len(PBS3.load)) + '
TEUs\n',
                             x=50, y=600, fontsize=14)

# # PBS1
# sim.AnimateMonitor(PBS1.load.length, x=60, y=600, width=200, height=25,
#                    horizontal_scale=1/500, vertical_scale=1/100,
#                    title=lambda: '#1 Location: ' + str(PBS1.terminal) +
'\n     Load: ' + str(len(PBS1.load)) + ' TEUs',
#                    titlefontsize=12)
# # PBS2
# sim.AnimateMonitor(PBS2.load.length, x=60, y=545, width=200, height=25,
horizontal_scale=1/500,
#                    vertical_scale=1/100,
#                    title=lambda: '#2 Location: ' + str(PBS2.terminal) +
'\n     Load: ' + str(len(PBS2.load)) + ' TEUs',
#                    titlefontsize=12)
# # PBS3
# sim.AnimateMonitor(PBS3.load.length, x=60, y=490, width=200, height=25,
horizontal_scale=1/500,
#                    vertical_scale=1/100,
#                    title=lambda: '#3 Location: ' + str(PBS3.terminal) +
'\n     Load: ' + str(len(PBS3.load)) + ' TEUs',
#                    titlefontsize=12)

#
```

```
# sim.AnimateText(text=lambda: 'Non-performance rate: '+"{0:.2f}".format(
#     env.non_performance_count * 100 / env.containers_generated)+' %',
x=500, y=550, fontsize=16)
#
# sim.AnimateText(text=lambda: 'Number of containers generated:
'+"{0:.0f}".format(
#                   env.containers_generated), x=500, y=540, fontsize=12)

sim.AnimateMonitor(monitor_non_performance_rate, x=50, y=50, width=900,
height=400, horizontal_scale=1/100,
                   vertical_scale=2, title=lambda:'Non-performance rate:
'+"{0:.2f}".format(
    env.non_performance_count * 100 / env.containers_generated)+'
%\nNumber of containers generated: '+"{0:.0f}".format(
                  env.containers_generated))

# late_delivery = [Terminal1.late_delivery, Terminal2.late_delivery,
Terminal3.late_delivery]
# ontime_delivery = [Terminal1.ontime_delivery, Terminal2.ontime_delivery,
Terminal3.ontime_delivery]

# sim.AnimateText(text=lambda: 'Late delivery / Total delivery: ' +
"{0:.2f}".format(
#     sum(late_delivery) / sum(late_delivery) + sum(ontime_delivery)) + '
%', x=50, y=500, fontsize=12)

## starting the simulation environment ##

env.run(env.weeks(6)) # run for a year

## saving .csv and create graph

non_performance_df = pd.DataFrame(np.array(
    monitor_non_performance_rate.xt()).transpose(), columns=['non-
performance (%)','time (min)'])
non_performance_df.to_csv(datetime.datetime.now().strftime('%Y-%m-%d
%H:%M')+' _nonperformance.csv')
ax = sns.lineplot(x="time (min)", y="non-performance (%)",
data=non_performance_df)
plt.savefig(datetime.datetime.now().strftime('%Y-%m-%d %H:%M')+'
_nonperformance.png')
plt.gcf().clear()

SatS_quaycraneQ_df = pd.DataFrame(np.array(
    SpaceAtSeaPlatform.quaycraneQ.length.xt()).transpose(),
columns=['number of available QC', 'time (min)'])
SatS_quaycraneQ_df.to_csv(datetime.datetime.now().strftime('%Y-%m-%d
%H:%M')+' _quaycrane.csv')
ax = sns.lineplot(x="time (min)", y='number of available QC',
data=SatS_quaycraneQ_df)
plt.savefig(datetime.datetime.now().strftime('%Y-%m-%d %H:%M')+'
_quaycrane.png')
plt.gcf().clear()

SatS_handlingtime_df = pd.DataFrame(np.array(
    SpaceAtSeaPlatform.vesselQ.length_of_stay.x()).transpose())
SatS_handlingtime_df.to_csv(datetime.datetime.now().strftime('%Y-%m-%d
%H:%M')+' _handlingtime.csv')
ax = sns.distplot(SatS_handlingtime_df, kde=False)
```

```
plt.savefig(datetime.datetime.now().strftime('%Y-%m-%d %H:%M')+'
_handlingtime.png')
plt.gcf().clear()

SatS_stack_df = pd.DataFrame(np.array(
    SpaceAtSeaPlatform.stack.length.xt()).transpose(), columns=['number of
containers (TEUs)', 'time (min)'])
SatS_stack_df.to_csv(datetime.datetime.now().strftime('%Y-%m-%d %H:%M')+'
_stack.csv')
ax = sns.lineplot(x="time (min)", y='number of containers (TEUs)',
data=SatS_stack_df)
plt.savefig(datetime.datetime.now().strftime('%Y-%m-%d %H:%M')+'
_stack.png')
plt.gcf().clear()

SatS_dwelltime_df = pd.DataFrame(np.array(
    SpaceAtSeaPlatform.stack.length_of_stay.x()).transpose())
SatS_dwelltime_df.to_csv(datetime.datetime.now().strftime('%Y-%m-%d
%H:%M')+' _dwelltime.csv')
ax = sns.distplot(SatS_dwelltime_df,kde=False)
plt.savefig(datetime.datetime.now().strftime('%Y-%m-%d %H:%M')+'
_dwelltime.png')
plt.gcf().clear()


## saving trace output to a file

save_stdout = sys.stdout
sys.stdout = open(datetime.datetime.now().strftime('%Y-%m-%d %H:%M')+'
_outputstatistics.txt', 'w')

print('================================\nNon-performance
rate:\n================================\n\n'+"{0:.2f}".format(
    env.non_performance_count * 100 / env.containers_generated) + '
%\n\n'+"{0:.2f}".format(
    env.non_performance_count * 100 / (

env.non_performance_count+Terminal1.ontime_delivery+Terminal2.ontime_deliv
ery+Terminal4.ontime_delivery
            +
Terminal5.ontime_delivery+Terminal6.ontime_delivery+SpaceAtSeaPlatform.ont
ime_delivery))+' %\n\n')

print('================================\nTerminal
1:\n================================\n\n')

print("On-time: " + "{0:.0f}".format(Terminal1.ontime_delivery) + '\nLate
(import): ' + "{0:.0f}".format(
    Terminal1.late_delivery) + " (" +
"{0:.0f}".format(Terminal1.late_import) + ')\nMean dwell time: ' +
"{0:.2f}".format(
    Terminal1.stack.length_of_stay.mean() / 1440) + ' days\n\n')

Terminal1.vesselQ.print_histograms()
Terminal1.stack.print_histograms()
Terminal1.quaycraneQ.print_histograms()

print('\n\n================================\nTerminal2:\n==============
==================\n\n')
```

```
print("On-time: " + "{0:.0f}".format(Terminal2.ontime_delivery) + '\nLate
(import): ' + "{0:.0f}".format(
    Terminal2.late_delivery) + " (" +
"{0:.0f}".format(Terminal2.late_import) + ')\nMean dwell time: ' +
"{0:.2f}".format(
    Terminal2.stack.length_of_stay.mean() / 1440) + ' days\n\n')

Terminal2.vesselQ.print_histograms()
Terminal2.stack.print_histograms()
Terminal2.quaycraneQ.print_histograms()

print('\n\n==================================\nTerminal3:\n==============
==================\n\n')

print("On-time: " + "{0:.0f}".format(Terminal3.ontime_delivery) + '\nLate
(import): ' + "{0:.0f}".format(
    Terminal3.late_delivery) + " (" +
"{0:.0f}".format(Terminal3.late_import) + ')\nMean dwell time: ' +
"{0:.2f}".format(
    Terminal3.stack.length_of_stay.mean() / 1440) + ' days\n\n')

Terminal3.vesselQ.print_histograms()
Terminal3.stack.print_histograms()
Terminal3.quaycraneQ.print_histograms()

print('\n\n==================================\nTerminal4:\n==============
==================\n\n')

print("On-time: " + "{0:.0f}".format(Terminal4.ontime_delivery) + '\nLate
(import): ' + "{0:.0f}".format(
    Terminal4.late_delivery) + " (" +
"{0:.0f}".format(Terminal4.late_import) + ')\nMean dwell time: ' +
"{0:.2f}".format(
    Terminal4.stack.length_of_stay.mean() / 1440) + ' days\n\n')

Terminal4.vesselQ.print_histograms()
Terminal4.stack.print_histograms()
Terminal4.quaycraneQ.print_histograms()

print('\n\n==================================\nTerminal5:\n==============
==================\n\n')

print("On-time: " + "{0:.0f}".format(Terminal5.ontime_delivery) + '\nLate
(import): ' + "{0:.0f}".format(
    Terminal5.late_delivery) + " (" +
"{0:.0f}".format(Terminal5.late_import) + ')\nMean dwell time: ' +
"{0:.2f}".format(
    Terminal5.stack.length_of_stay.mean() / 1440) + ' days\n\n')

Terminal5.vesselQ.print_histograms()
Terminal5.stack.print_histograms()
Terminal5.quaycraneQ.print_histograms()

print('\n\n==================================\nTerminal6:\n==============
==================\n\n')

print("On-time: " + "{0:.0f}".format(Terminal6.ontime_delivery) + '\nLate
(import): ' + "{0:.0f}".format(
```

```
    Terminal6.late_delivery) + " (" +
"{0:.0f}".format(Terminal6.late_import) + ')\nMean dwell time: ' +
"{0:.2f}".format(
    Terminal6.stack.length_of_stay.mean() / 1440) + ' days\n\n')

Terminal6.vesselQ.print_histograms()
Terminal6.stack.print_histograms()
Terminal6.quaycraneQ.print_histograms()

print('\n\n==================================\nSpace@Sea:\n==============
==================\n\n')

print("On-time: " + "{0:.0f}".format(SpaceAtSeaPlatform.ontime_delivery) +
'\nLate (import): ' + "{0:.0f}".format(
    SpaceAtSeaPlatform.late_delivery) + " (" +
"{0:.0f}".format(SpaceAtSeaPlatform.late_import) + ')\nMean dwell time: '
+ "{0:.2f}".format(
    SpaceAtSeaPlatform .stack.length_of_stay.mean() / 1440) + ' days\n\n')

SpaceAtSeaPlatform.vesselQ.print_histograms()
SpaceAtSeaPlatform.stack.print_histograms()
SpaceAtSeaPlatform.quaycraneQ.print_histograms()

print('\n\n==================================\nPBS:\n====================
=============\n\n')
PBS1.load.print_histograms()
PBS2.load.print_histograms()
# PBS3.load.print_histograms()
```

Appendix F NEN 2018: Weather effect on the load scenarios of a quay crane

Available at :
https://connect.nen.nl/standard/openpdf/?artfile=367380&RNR=7471&token=1ebdcd65-cb13-4dcb-aed1-07b7eb96047f&type=pdf#pagemode=bookmarks [Accessed: 24-Sep-18]

Note
Although this type of crane does not conform to the definition given in NEN 2017, it is included in this specification as an exception.

4.6 Load resulting from weather effects

4.6.1 General

The following effects should be distinguished:
a. wind and storm; effect S_W and S_S respectively;
b. snow; effect S_N;
c. temperatur; effect S_{TE}.

4.6.2 Wind; effect S_W and storm; effect S_S

It is supposed that the effect of the wind is horizontal. The wind direction most unfavourable for a crane must be taken into account.

4.6.2.1 Wind pressure

The wind pressure q is determined by the formula:

$$q = \tfrac{1}{2} \rho v_W^2 = \frac{v_W^2}{1,6}$$

in which:
q is the wind pressure in N/m^2
v_W is the wind velocity in m/s
ρ is the density of the air, to be specified at $\frac{1}{0,8}$ kg/m^3.

Table 20 gives values for q and v_W; table 21 states the wind scale and the appertaining wind pressures.

Table 20 -- Load through wind and storm on crane parts

type of crane	height above soil in m	WIND crane in operation			STORM crane out of operation		
		v_W		q	v_W		q
		m/s	km/h	N/m^2	m/s	km/h	N/m^2
A							
cranes that are easily secured against wind action, and are designed for operation in light wind only (for example cranes on a low frame height with booms that can easily be lowered to the ground)	0-20	14	50	125	--	--	--
	20-100	14	50	125	--	--	--
	> 100	14	50	125	--	--	--
B							
all normal types of cranes installed in the open, with the exception of types A and C	0-20	20	72	250	36	130	800
	20-100	20	72	250	42	150	1100
	> 100	20	72	250	46	165	1300
C							
type of crane which must continue to work in high winds	0-20	28,5	102	500	36	130	800
	20-100	28,5	102	500	42	150	1100
	> 100	28,5	102	500	46	165	1300

Regarding the operation with these cranes, see NEN 2024 and NEN 2026.
For the loads by wind and storm on crane parts, normally the values must be regarded as recorded for kind B, unless the classification according to another type is possible or necessary and is agreed between the concerned parties.

Table 21 -- Wind scale and appertaining wind pressures (q)

wind force 1)	wind speed averaged over 10 min at 10 m height above flat ground or above sea		name		wind pressure q N/m^2
	m/s 1)	km/h 2)	above flat ground	above sea	
0	0,0- 0,2	0,0- 0,7	calm	calm	0 - 0,03
1	0,3- 1,5	1,1- 5,4	light wind	flat and calm	0,06- 1,4
2	1,6- 3,3	5,8- 11,9		flat freshening	1,6 - 6,8
3	3,4- 5,4	12,2- 19,4	moderate wind	slight freshening	7,2 - 18,2
4	5,5- 7,9	19,8- 28,4		moderate fresh.	18,8 - 39
5	8,0-10,7	28,8- 38,5	fairly strong wind	fresh breeze	40 - 71
6	10,8-13,8	38,9- 49,6	strong wind	stiff breeze	73 -118
7	,9-17,1	50 - 61,5	hard wind	hard wind	121 -182
8	17,2-20,7	61,9- 74,5	stormy wind	stormy wind	184 -267
9	20,8-24,4	74,9- 87,8	storm	storm	270 -371
10	24,5-28,4	88,2-102,2	heavy storm	heavy storm	375 -502
11	28,5-32,6	102,6-117,3	very heavy storm	very heavy storm	509 -662
12	>32,6	>117,3	hurricane	hurricane	>662

1) The figure for the wind force is borrowed from the International Beaufort scale, which is originally defined above sea. Depending on the dimensions of waves and the presence of foam crests and suchlike the wind velocity can be estimated with the help of this scale.
Above land therefore it is more difficult, and inaccurate, to work with this scale.
The indication of the wind force, and the relation with the wind velocity at present to be measured in m/s, however are applied internationally both above land and above sea for the classification of the wind.
2) The values for the wind velocity in km/h are deduced from those in m/s.

4.6.2.2 Calculation of the wind effect

The force exerted by the wind on a construction element or a girder in the direction of the wind amounts to:

$$F_W = A \cdot q \cdot C$$

in which:
F_W is the wind force, in N
A is the sum of the projections of the surfaces visible and encountered in the direction of the wind in m^2 (projected on a surface perpendicular to the direction of the wind)
q is the wind pressure, in N/m^2
C is an aerodynamic coefficient, which depends on the shape of the surface concerned. Values of C have been given in table 22. (See also figure 10.)
More favourable values may be adopted, if these are proved by means of wind tunnel tests.

4.6.2.4 Wind load on the hoisting load, effect S_{WL}

For the effect of the wind load on the load the greatest value which can be determined on the basis of the known or anticipated surface and the shape of the load must be taken into account.
At least the following must be retained:
a. for cranes of type A according to table 20:
$F_{WL} = 0,015 \cdot L_n \cdot g$ with a minimum of 250 N
b. for cranes of type B according to table 20:
$F_{WL} = 0,03 \cdot L_n \cdot g$ with a minimum of 500 N
c. for cranes of type C according to table 20:
$F_{WL} = 0,06 \cdot L_n \cdot g$ with a minimum of 1000 N
in which:
L_n is the effective load, in kg
g is the strength of the local gravitational field, to be put at 10 N/kg.

4.6.3 Snow; effect S_N

Load through snow does not need to be taken into account in general. In cases where great horizontal surfaces or snow drifts and suchlike may lead to relatively large loads, a load of 500 N/m^2 should however be retained.

4.6.4 Temperature; effect S_{TE}

Stresses, caused by temperature change or temperature differences need not to be taken into account in general, unless in consequence relatively large contributions to the stresses are caused (as a guid line more than 5 % of the total stress).
For cranes, set up in the open air in the Netherlands, fluctuations in the temperature of the atmosphere between -20 °C and +40 °C and with temperature differences of 15 °C through unequal heating of separate parts should be taken into account.
For cranes which are exposed to differing simultaneously occurring ambient temperatues, temperature differences adapted to these special circumstances must be considered.
The coefficient of linear expansion for steel amounts to:

$$\alpha_T = 0,000\ 012\ K^{-1}$$

K^{-1} = per Kelvin.

4.7 Load from cabins and accesses; effect S_{KT}

4.7.1 Cabins, stairs, gangways and landings must be calculated at a movable point load of 3000 N or a uniform load of 1500 N/m^2.
The largest of these 2 loads must be taken into account. In cases where a larger load is to be expected this must be taken into consideration.
Ladders must be calculated at a movable point load of 1500 N.
Hand rails must be calculated at a horizontal movable point load of 500 N acting perpendicularly to the hand rail.

4.7.2 The effect of the load on accesses (stairs, ladders, gangways and landings) need not to be taken into account in the calculation of the main supporting construction.

Appendix G A rough comparison of Salabim to TOMAS

Obtained from the mailing group of Salabim users (salabim@googlegroups.com)

Please find below a comparison of salabim and Tomas, two DES packages.

This overview is doubtless biased (the author is the core developer of salabim) and may contain incorrect
or incomplete information. Other contributors, particularly Tomas users and developers, are invited to update
the information given below.

Unavoidably, this overview is also a comparison between Delphi/Pascal and Python.

General
=======
The DES packages salabim and Tomas are quite similar, not in the least because they are both more or less derived
from Must (by Ruud van der Ham, the author of salabim).

Basic process functionality
===========================
The basic process functionality is comparable with some terminology differences:

salabim	Tomas
yield activate for current component	N/A
activate for other component	Resume, Start
yield passivate for current component	Suspend
passivate for other component	N/A
yield hold for current component	Hold
hold for other component	N/A
yield cancel for current componen	Finish, FinishAndDestroy
cancel for other component	Cancel
interrupt (stacked)	Interrupt, Pause (not stacked?)
yield standby, standby	Standby

Extended process functionality
==============================
Both salabim and Tomas support resources, although salabim supports reneging and the claimers and
requesters and claimers queues are just standard queues with all advantages of monitoring and animating.
Salabim also contains so called anonymous resources, that are not present in Tomas.

On top of that, salabim has a very powerful State class which allows a condition to be checked (wait)
without the overhead of standby.

Queue handling
==============
Basic queue handling is similar, with different terminology.
As queues are handled as a standard 'ABC class', a very rich idiom is present.
For instance, looping over (all) elements in a queue is more intuitive in salabim:

salabim	Tomas

for ship in arrivals:
  ...
	Ship:=Arrivals.FirstElement;
	While Ship<>Nil Do
	  Begin
	  ...
	  Ship:=Arrivals.Successor(Ship)
	  End;

Also in salabim, several queries can be done without a call, like:

```
    if c in arrivals:
```

or

```
    first = arrivals[0]
```

, but the more conventional, Tomas like constructs are still available.

In salabim the queue length and the length of stay in a queue are automatically monitored.
I am not sure if and how that works in Tomas.

The content of a queue can be animated in salabim with just one statement.
In Tomas that requires more work, as far as I know.

Monitors
========
Tomas' collections are similar to salabim's monitors and both packages support visualization on
a time scale.
Salabim has more options to get statistics, like percentiles and number of entries between a lowerbound
and upperbound.
Histograms are presented quite differently. Salabim still uses text histograms, that are fully customizable.
Tomas supports more modern graphical histograms, with less flexibility.
Salabim's collected time series can be easily exported to other (statistical or presentation) packages, like
matplotlib, numpy or pandas. I am not sure about Tomas.

Animation
=========
Salabim has an advanced, optionally realtime, 2D animation engine that can also be used to produce high
quality videos out of the box.
Tomas?

GUI
===
Tomas uses the advanced Delphi GUI components, which make it a snap to build nice forms and generate
high-quality output.
In salabim that is much more complicated, if at all possible.

Statistical sampling
====================
Salabim offers more statistical distributions to sample from.

Reliability
===========
I think both packages can be used to acquire reliable results.
The random generators can both provide reproducibility.

The trace functionality of salabim is more elaborate and even shows the line numbers. Therefore it is
arguably easier to validate a model in salabim.

Python is a dynamic, non-typed, language, which might lead to errors that are hard to find.
Delphi/Tomas, on the other hand, is fully typed and will detect some errors already at compile time.

Speed
=====
Execution speed in Tomas is superior to salabim, due to the host language.
In the Python ecosystem, there's an alternative runtime system, called PyPy that makes execution
much faster. Benchmarks with older versions of salabim showed that Tomas models run appr. 2 times
faster under PyPy.

Development time is another issue. I personnally think that Python is superior in that respect,
not in the least by the availability of sophisticated IDEs, debugging and testing tools.

Also there is much more material for Python than Delphi/Pascal to learn the language and environment.

Other aspects
==========
Tomas is available under a commercial (relatively expensive) Delphi license, a free community,
restricted Berlin license or the open source Lazarus project. It runs under Windows, OSX and Linux.

Python is fully open source, free and available under Windows, OSX, Linux and iOS. Therefore, salabim models
can be even be developed and run on iPad/iPhone !

Salabim is released under the MIT license and is fully open source.
Tomas license conditions are not very clear (at least to me). It is for sure not fully open source, as of now.

Salabim has a very active user group and offers (free or commercial) support options.
Tomas ?

Python an extremely large and nearly fully open source library (machine learning, web interface, database,
statistics, graphics, I/O, etc.). Delphi?
Compared to Delphi, Python has far more developers and users, which might make it easier to find developers and testers.

Finally
=======
Experience, personal preferences and specific needs will for sure influence the choice of a Discrete Event
Simulation package.
Please observe that there are several other DES packages available:
- SimPy, under Python with a quite different API and rather limited functionality
- SimJulia, like SimPy under Julia
- Simmer, like SimPy under R
- DSOL, a not very well maintained package under Java.
- ...

On Friday, 5 October 2018 11:26:25 UTC+2, **** wrote:
Dear Ruud,

During my studies I worked with the Tomas simulation package created by H. Veeke. However, I have more experience with programming in Python and would therefore like to use your Salabim package for my graduation project. How do you compare the two packages on functionallity, reliability and speed?

Best, ***
--