20th International Conference on Modelling and Applied Simulation
18th International Multidisciplinary Modeling & Simulation Multiconference

ISSN 2724-0037 ISBN 978-88-85741-61-4 © 2021 The Authors.
doi: 10.46354/i3m.2021.mas.002

A model-driven design approach for Ro-Ro and container terminals: from requirements analysis down to simulation model implementation

Mohamed Nezar Abourraja[1,*], Sebastiaan Meijer[1] and Jaouad Boukachour[2]

[1]KTH Royal Institute of Technology, Stockholm, Sweden
[2]Normandie University, UNIHAVRE, 76600 Le Havre, France

[*]Corresponding author. Email address: mnabour@kth.se

Abstract

Modeling, one of the main pillars of good scientific research, is a long-standing multidisciplinary activity to understand and analyze complex systems. In this paper, the focus is directed toward conceptual modeling of multi-terminal seaports specialized in handling and treatment of intermodal transport units (ITU). These systems are complex with highly dynamic and stochastic behaviors and actors, therefore, studying them as a coherent whole or just analyzing one part by taking into account the high degree of integration among the different aspects and actors linked by a flow of activities, information, and interactions is a bet lost in advance without a well-defined design process. Several design approaches and methodologies have been proposed over the years, but nonetheless, there is still no agreement on how to conduct modeling of complex systems because they are of different kinds. In this line, this paper proposes a top-down approach for container and Ro-Ro terminals largely inspired by the Unified Process Methodology and refined through several research projects that we have been involved in. It gives some recommendations and guidelines as well as a helpful way to successfully build modular and consistent simulation models. To prove its efficiency, it was applied to a case study and the resulting models were validated by the subject matter's experts.

Keywords: Complex system, Modeling approach, Model-driven development, Simulation modeling, Multi-agent system

1. Introduction

Modeling is, above all, an art before being a science, where creativity, knowledge, and experience play a significant role. While much has already been written on this topic, there is still no agreement on how to conduct modeling of complex systems. This is understandable because actually complex systems are of a different nature and structure as well as the purposes behind modeling being many. However, designing models is no mean feat, and a roadmap for modeling needs to be defined first. To this end, this paper proposes a top-down approach consisting of four steps that starts from the requirement analysis down to the simulation model implementation through a set of artifacts and diagrams. This design approach could be seen as an instance of the Unified Process (UP), and is specially defined for Ro-Ro and container terminals, but it might be applied to other system types depending on how similar they are to the studied systems.

1.1. General background

For many decades, complex systems have constantly attracted the attention of the scientific community as they are a source of complicated problems. Generally speaking, a system is designated as complex when its behavior is intrinsically difficult to predict due to the high interconnections among its components evolving in

© 2021 The Authors. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY-NC-ND) license (https://creativecommons.org/licenses/by-nc-nd/4.0/).

a stochastic environment toward achieving individual and/or collective objectives. Hence the interest in investigating such systems to come up with appropriate solutions for complicated problems thereof.

Here, the focus is on the modeling and analyzing of maritime multi-terminal seaports specialized in handling and treatment of intermodal transport units (ITU) (see Figure 1). A multi-terminal seaport is a well-structured and sensitive system where wheeled and containerized freights are subjected to multiple processing and handling operations before being delivered to their outgoing transportation modes. Usually, these operations are performed within an uncertain and vulnerable environment, and sometimes under a lack of information needed for sound planning; consequently undesirable situations could arise. The stochastic aspect of maritime terminals stems also from unforeseen perturbations and risks arising after human errors or other uncontrollable factors.

To come up with consistent and relevant models for such systems, the appropriate modeling approach should be selected in terms of the studied system characteristics as well as the purposes behind the study. According to Günther and Kim (2005) and Garro and Russo (2010), conceptual modeling and multi-agent systems (MAS) are promising and suitable approaches for designing and analyzing logistic systems and have easily found their way to container terminal applications. Combining both of them gives birth to so-called multi-agent-based simulation models (MABS). With such models, individualities and emergent phenomena can be easily captured in addition to the collective behaviors, which give a closer image of reality and thus lead to a thorough understanding of system functioning.

Figure 1. Norvik multi-terminals seaport: Ro-Ro and containers terminal

1.2. Issue under focus

Going from an informal description that could be misunderstood to a well-formalized representation of a complex system is in itself a puzzle; that is, a roadmap to simplify the modeling process is strongly needed. As human beings, we can spontaneously design in the mind very simple representations of systems to face daily situations (i.e., forming mental images of systems) without the need to a modeling process to make it, but when more details and complexity are considered, such as interactions, environmental impacts, behaviors, system composition and evolution, constraints, etc., it is easy to get lost along the way in modeling or even not know where to start. Indeed, through the help of a modeling approach and using the proper level of detail, a careful analysis of both the quantitative and qualitative aspects of the studied system could be carried out.

In the literature (Bresciani et al. 2004; Garro and Russo 2010; and Fortino and Russo 2012), there is no unanimity on a specific modeling process to build models; however, from our point of view, we believe that the conception of models may depend on three influencing factors: system designer awareness and perspective, the nature of the studied system and the level of abstraction.

In fact, system designers can come up with different models for the same system relying not only on their experience but also on their creative spirit. Moreover, complete knowledge about system functioning is essential for a smooth transition process, and any lack of that obliges the designer to make use of abstraction and assumptions to build consistent designs. Besides, the structure of any design is usually established according to the purposes for which the system is being modeled. For instance, business management software generally follows the MVC architectural pattern, while simulation models are usually based on Discrete Event System Specification. The structure of the model also relies on the studied system itself. Each system has its own characteristics, composition, and aspects, so the model should comply as much as it is potentially possible with these elements to be a realistic image of its corresponding system.

Although the model is designed to act as much as possible as the reference system, it generally represents only a part of reality because of abstraction. The abstraction is applied to hide out of scope aspects (narrowing the scope of the study) or a lack of knowledge as well as to reduce system complexity in order to catch only essential elements for the study's purposes. However, the abstraction has a significant impact on the results' accuracy as the more the abstraction increases, the more the accuracy of the results decreases. On the other hand, the lower the abstraction level, the more complex and time-consuming the designing and modeling process. Thus, the choice of the suitable abstraction level is a reply to the following questions:

- What is the designer looking for? (i.e., designer's purposes).
- What knowledge is available about system functioning? (i.e., knowledge availability).

Therefore, to build a consistent virtual representation for complex systems, and particularly for Ro-Ro and container terminals, this paper

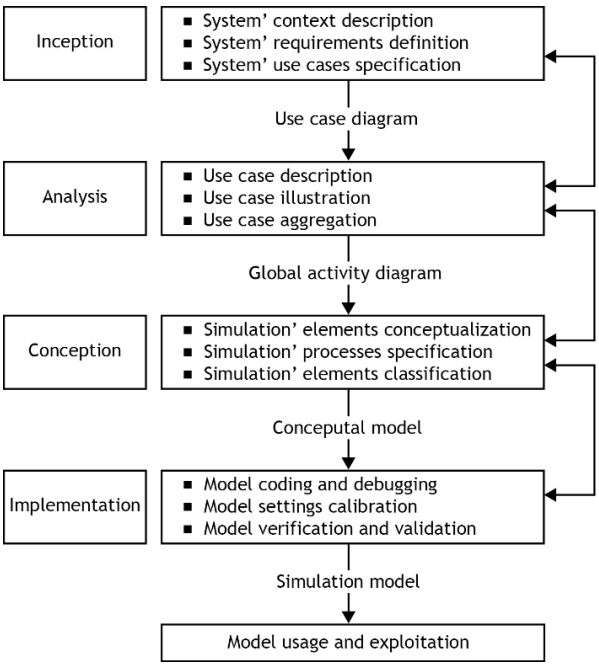

Figure 2. The proposed design approach

introduces a design approach composed of four steps that starts with a requirements analysis down to a simulation model implementation through a set of linking diagrams and artifacts. This approach is applied to the Norvik seaport terminals (see Figure 1). While the main steps are already specified in the literature (Kruchten 2004; Garro and Russo 2010; and Fortino and Russo 2012), the modeling activities in each step and the transition mechanisms linking the steps to one another are still a matter of debate among the scientific community. This paper intervenes in this debate by proposing a model-driven design approach for Ro-Ro and container terminals. It gives some recommendations and guidelines to successfully perform each step as well as the sub-steps of modeling to facilitate transitions between the steps in order to end with consistent simulation models for container and Ro-Ro terminals.

The remainder of this paper is organized as follows. The next section highlights the proposed approach, sections 3 to 6 illustrate in depth each step of our approach, successively. The last section discusses the approach and concludes the paper.

2. Proposed modeling approach: an overview

In line with the aforementioned factors, we defined a top-down approach (see Figure 2), with a set of steps: inception, analysis, conception, and implementation. Most of the reviewed approaches can be staged into these four steps. Bresciani et al. (2004) introduced an agent-based methodology for software development (Tropos) inspired from the well-known methodology "Unified Process" (Kruchten 2004) while adapting its artifacts to design agent-based software. Garro and Russo (2010) proposed a methodology to design MABS models for complex systems. The methodology contains six steps: system analysis, conceptual system modeling, simulation design, simulation code generation, simulation set-up and simulation execution, and results analysis. The first step is about requirement definition (inception) and analyzing of system behavior. The second and third steps concern the conception of the simulation model. From the fourth step until the last one, the implementation and model testing are carried out. Kubera et al. (2011) illustrated an approach divided into four steps called "Interaction-Oriented Design of Agent simulations" (IODA) to build simulation models. This approach focuses mainly on modeling agents and their interactions in complex environments. In Fortino and Russo (2012), an agent-based methodology of three phases named "ELDAMeth" was introduced. Here, the first phase includes the inception and system analyzing activities. The last phases are about conception and implementation, respectively. Other approaches and methodologies have been proposed; interested readers can see: Gaia (Wooldridge et al. 2000), Prometheus (Padgham and Winikoff 2002), etc. At the end, all of these approaches provide guidelines, recommendations, and tools to build gradually agent-based models from scratch.

Our approach goes in this direction; however, it is interested especially in making models for Ro-Ro and contains terminals. But we believe that the presented material might be applied to other system types depending on how similar they are to the studied systems in this paper. As illustrated in Figure 2, the process of development ensures a progressive conceptualization of the complex system through a set of formal diagrams linking the steps to each other to end with a well-defined representation. The forward and backward transitions over the steps, in case of incomplete or erroneous details, allow the model to be further refined and more realistic. For example, in the conception step one can notice that certain insights included in the previous step are not enough to go further in coding the models. In the first step, a very abstract image of the system is obtained, then more details and insights are captured in the second step to end with a thorough understanding of the system's functions so that system components can be conceptualized in terms of agents, objects, processes, messages, etc. to build a conceptual virtual presentation, i.e., a model, of the reference system that should be easy to implement. It should be noted that the sub-steps in each step could be achieved in a parallel or sequential manner.

The steps of this approach are mainly established and defined following the "Unified Process" methodology (Kruchten 2004). The sub-steps are inspired by the reviewed approaches and are a concretization of gained experience and feedback through several conducted research projects on building simulation models for port terminals: (1) the hinterland terminal of Le Havre seaport (Abourraja et al. 2017; Abourraja et al. 2018; Rouky et al. 2018); (2) a rail-road container terminal

(Abourraja et al. 2019; Benantar et al. 2020); The main novelty of this approach is introduced in Section 5.

3. Step I: Inception

This initial step, also called requirement analysis, is the basis of our design approach with the aim of answering "who" and "what" questions regarding our studied systems. Here, the issue is not about drawing a holistic picture on system functioning and composition, but rather about collecting the details and knowledge needed to design a model that fits with the study´s objectives. To this aim, first of all, the objectives and purposes are enumerated, then the system context and boundaries are delimitated to identify relevant actors and components, and finally the functional and non-functional requirements deserving attention are captured.

At this early stage of modeling, the designer has only a fuzzy picture on the studied system and is unaware of which aspects are worthy of interest. Thus, before starting, it is of paramount importance to examine the available documentation either in the literature or that provided by system' stakeholders in addition to having a meeting with them and on-the-spot visits to the studied systems. With these activities, a general idea and enough knowledge on the studied system with insights on research progress and trends could be acquired to move further with the modeling. Despite that, some collected facts might be ambiguous or incomplete, which is quite comprehensible since they are incrementally refined, adjusted, and extended as the development steps and iterations are carried out. The key concepts in this step are: actors, entities, use cases, and objectives. As for functional and non-functional requirements, they specify the individual functions of the system (i.e., what the system does) and the recommendations to operate those functions (i.e., how the system should do it), respectively. As a bridge for the next step, the functional and non-functional requirements are clearly specified in terms of use cases and mapped to their respective actors to make up the use case diagram.

3.1. Objectives and assumptions

The intrinsic purpose of simulation models is obviously to mimic the behavior of real systems narrowed to a proper level of realism. More precisely, the degree of the represented reality in simulation models is seen as one of the keys to success, especially from the end-user's point of view. However, representing some parts of reality could be meaningless to the designer because they have no impact on the desired outputs. Thus, operations and actors that have no relation to the planning and execution of processing and handling operations inside the system are not considered (e.g., food and energy suppliers, firefighters, cleaners, security services, etc.). Furthermore, to investigate accurately the performance of the system, a microscopic representation of the studied systems must be designed; that is, a low level of abstraction is chosen. On the other hand, some complex behaviors could be aggregated into more simple forms while maintaining the realism of the model in order to reduce the complexity of the design. Thus, driver behaviors and physical phenomena (road and air friction) are defined as probability distributions of times and speeds of handling and transportation equipment. Moreover, the risks are regular (delays, failures, interference, collisions, congestion, etc.) or irregular (natural disasters, explosions, leakages of liquids or gas, etc.). The last risk type is excluded.

Otherwise in this study, the following assumptions are made:

- Ship-train stowage plans are assumed to be known.
- Outgoing trucks for some import ITUs might be unknown when they arrive at the terminal from the sea; this is specified a few hours before the arrival of trucks.
- Reachstacker scheduling comes before the scheduling of straddle carriers of the inland pool.

3.2. System context and description

The maritime terminal's operator plans handling operations according to the upstream information received from shipping lines and inland transporters. This information contains detailed descriptions on incoming flows. Sometimes, false declarations of ITU contents could be made to hide illicit freight intended to unlawful activity. To thwart such illegal activity, terminal and port authorities, customs services, and other governmental agencies work together to identify suspicious ITUs to be inspected by customs in a dedicated area in the terminal using X-ray scanners and/or manual inspection.

There are three types of flows. First, the import flow, also called inbound flow, is received from the sea, then routed to landside to be evacuated to the hinterland destination. Conversely, the export flow (outbound flow) is collected from trains and trucks to be subsequently loaded on or into sea-going ships. The last type is a particular flow labeled "transit flow" that is unloaded from ships and loaded on to or into other ships (i.e., transshipments among same-type modes). The coordination of these flows is the role of transport service providers as the major actors of the supply chain, which also schedule transportation means' round trips.

A slight summary on maritime terminals was given in the Introduction (see Figure 1). From a physical viewpoint, the studied system is composed of two terminals: the container part and ro-ro part. Each terminal is a set of interconnected operating areas, namely seaside, internal yard, and inland-side. These areas are interrelated by a road network. The inland-side of the container part differs from that of the ro-ro part. The first one is subdivided into two sub-areas, rail-side and road-side, whereas the second one has only a road-side. The seaside, commonly called quay

side, is where ships are moored to berths, then handled by quay cranes in the case of containers and unloaded by internal tractors via linkspans in the case of trailers. Additionally, certain external trucks (not detachable trucks, here named lorries) can drive on to or off ships on their own. The landside is a zone where operations on trucks and trains take place. In the road-side, both ro-ro trucks and container trucks enter and leave the terminal by entry and exit gates. Moreover, incoming full trucks are first oriented to the security check hall for security and weight checking before accessing their respective terminal. Container trucks are unloaded by straddle carriers within an area arranged for that purpose near the gates (XT-handover area). Regarding trains, they are lined up on parallel tracks at the rail-side and processed by reachstackers. Finally, in the internal yard, straddle carriers temporarily pile up containers in blocks whilst internal tractors line up inbound trailers in parking lots. Physically, blocks are interspaced areas consisting of a set of interspaced lines composed of stacks with n tiers, whereas parking lots are interspaced lines composed of spots. Concerning outbound trailers, they are dragged directly to parking lots by their external tractors.

From a functional viewpoint, maritime terminals are classified into four subsystems: the ship-to-shore subsystem (seaside operations), horizontal-transport subsystem (transport operations), storage subsystem (internal yard operations), and delivery-receipt subsystem (landside operations). The transport subsystem is the artery of the terminal as it connects all the components to each other, which gives rise to the problem of subsystems' synchronization; that is, poor synchronization between the transport subsystem and the other ones will slow down the performance of the whole system. These subsystems are governed by a set of decisions hierarchically structured into different levels. Firstly, the tactical level covers decisions about equipment deployment and seaside resource allocation. Secondly, the operational level includes decisions about day-to-day operations. Finally, the real-time level regroups very short-term decisions like equipment routing and storage management. In this paper, the strategic level is not considered since it concerns decisions about location and layout design problems.

As regards equipment and ITU, they are split into active and passive equipment (Stahlbock and Voß 2008), and movable and driven ITU, respectively. A decision's quality and performance is measured using key performance indicators (KPI). These KPI concern mainly times, moves, resource utilization, energy consumption, and costs (Kemme 2013). Other KPI could be considered. Furthermore, the performance of the terminal depends not only on the optimization of resources and equipment utilization but also on its customers' satisfaction.

3.3. Use case specification

A use case is either simple or composite, and those identified as simple are enclosed in composite ones. For instance, "unloading containers from/to ships" and "transferring containers to/from the shore" are embodied in "handling container ships". In the literature, two stereotypical relationships between use cases are reported (Kruchten 2004): include and extend relationships. The first one is used to indicate common parts of behaviors among several use cases. As an example, "handling container ships," "handling trains," and "container storing" included the use case "interference avoidance" (called internal case), because equipment always keep a safe distance from one another to avoid risky situations during handling operations. The second relationship is lighter than the first one and means that the use case could call other ones to extend its behavior. For instance, during ro-ro-ships loading, internal tractors leave trailers in pre-boarding areas if the queue to the ship is too long in order to bring the remaining trailers, so as to minimize unproductive waiting times. Likewise, in the case of lorries (not detachable trucks), trailers are positioned in parking spots without being decoupled from their tractors. There is another kind of relationship that could be stated which is quite similar to "include" except that it is implicit. In fact, handling and transport operations (equipment) are strictly based on decisions made at the scheduling and planning level (planner); however, they are carried out by different actors; besides, the execution of the former ones does not immediately involve the latter ones and vice versa. Therefore, it is safe to say that actors in charge of scheduling and planning will have control over actors executing handling tasks.

In regard to actors, they can cooperate and coordinate to handle a use case (e.g., fraudulent ITU targeting), or they can just be linked to the same use case without any particular relationships between them. For example, straddle carriers and trucks both transport containers but do not communicate to perform this task. Actors are people, equipment, or other systems as illustrated in the previous sub-sections. In this study, some actors are seen as one single unit in spite of there being more, because we have no interest in their individuality. This concerns the port authorities, terminal operators, labor, dockers, transport service providers, the customs service, governmental agencies, and simulation end users. The other actors are categorized into three major classes: handling equipment, transport equipment, and transport modes.

4. Step II: Analysis

At this stage of modeling, "how?," "why?," "when?," and "where?" questions are the main concern in order to investigate deeper the features and components of the studied system so as to reach a firm understanding of system functioning. To this aim, a meticulous and careful analysis of the defined use cases is the artery to determine the dependencies and relationships among the involved entities as well as the streams of the

executed actions. In this study, we consider an action as the smallest piece of treatment executed by an actor. This first sub-step of analyzing is translated into orthogonal and graphical views to gain more visibility following the dictum "a picture is worth a thousand words." As the fruit of labor of this step, the produced views in the second sub-step are aggregated to form the global model. This work-product is a preliminary simplified representation of the system which constitutes the foundation of the simulation model.

4.1. Use case unrolling

The analysis begins with a textual description organized in fields where the story of the use cases is explicitly and plainly told. Cockburn (2000) suggested some guidelines for breaking up use cases. In the first place, start with listing all the actors participating in any way in the use case. In the second place, give an overview on the mainstream of the use case (named basic flow), which is the common sequence of actions up to the goal. In the last place, write all alternative streams to the basic scenario while citing all exceptions, extensions, or events causing these deviations. The benefits of this practice are not only easy unrolling and situating of use cases but also the capture of redundant parts to induce other use cases, so the artifact of the previous step can be more refined. However, there is no standardized form for this exercise; text documents, text bubbles with annotations, tables, or other forms can be used. In this work, the choice fell on tables with eight fields (rows). To keep the length of this paper reasonable, the fields are enumerated and illustrated below instead of drawing up each table. Note that this is not exhaustive; some fields could be removed and others added (see Cockburn 1998; and Sindre and Opdahl 2001):

- ID-title: here, is about labeling and assigning an ID to use cases. For example, "1.road.gate.in.management," is the first process in the truck activity.
- Main actor/Secondary actors: these two fields reveal all the actors, whether the main actor, who is responsible for executing most of the actions of the use case, or secondary actors, who only participate in one of the actions. For instance, the planner orders the labor to decouple a train. In this case, the main actor is the labor, who perform the majority of the required actions, whereas the planner only sends the order and oversees the smooth running of the actions. The planner is the most involved actor in our system's use cases, followed by internal equipment either as main or secondary actors.
- Pre-conditions/Post-conditions: the first one are the conditions that ought to be satisfied to launch the use case. They refer to the state in which the system should be in beforehand. Unlike pre-conditions, they point out the system's state after the end of the use case. Both these fields make it possible to link the use cases together like a linked list, since the post-conditions of the prior use cases could be the pre-conditions for subsequent ones. Most pre-post-conditions are: arriving at the terminal, reception of the arrival notification, leaving the terminal, achieving a process or action, the emergence of phenomena (failures, repairs, delays, etc.), etc.
- Constraints: constraints are of two types: weak and strong. The strong constraints are the requirements to be respected during the execution of the use case, otherwise an exception is raised (blocking event). They are defined to ensure the correct behavior of the model (e.g., equipment-resource size and capacity, task precedence, time window, retrieved order, storage constraints, container type segregation, etc.) and to avoid risky situations (e.g., non-interference, ship stability, safe distances, speed limits, etc.). Sometimes one constraint becomes trivial in order to respect a stronger one. Regarding weak constraints, they do not represent any blockade for the system's processes, yet they should be taken into consideration for better performance of operations, i.e., synchronization of operations, maximizing resource utilization, evenly spread workload among resources and equipment, etc.
- Basic stream/Alternative streams: all possible scenarios should be spelled out along with the switching points between them. A scenario is a series of actions executed in a precise order, written in a simple and factual style. Each action can be associated to some events or conditions, and/or can undergo constraints in resource usage. As a rule, a use case has only one basic stream and could have multiple alternative streams.

Although a wealth of information is gained thanks to these descriptive fields, a picture is still worth a thousand words. Furthermore, the control flow elements (e.g., if-else conditions, loops, join and fork nodes, and transitions) are missing and parallel streams of actions are barely distinguishable in the textual description. Thus, to get enough visibility on use cases' execution, the scenarios are transformed into activity diagrams. In addition to enhancing visibility, this diagram highlights graphically the synchronization and connections between the use cases.

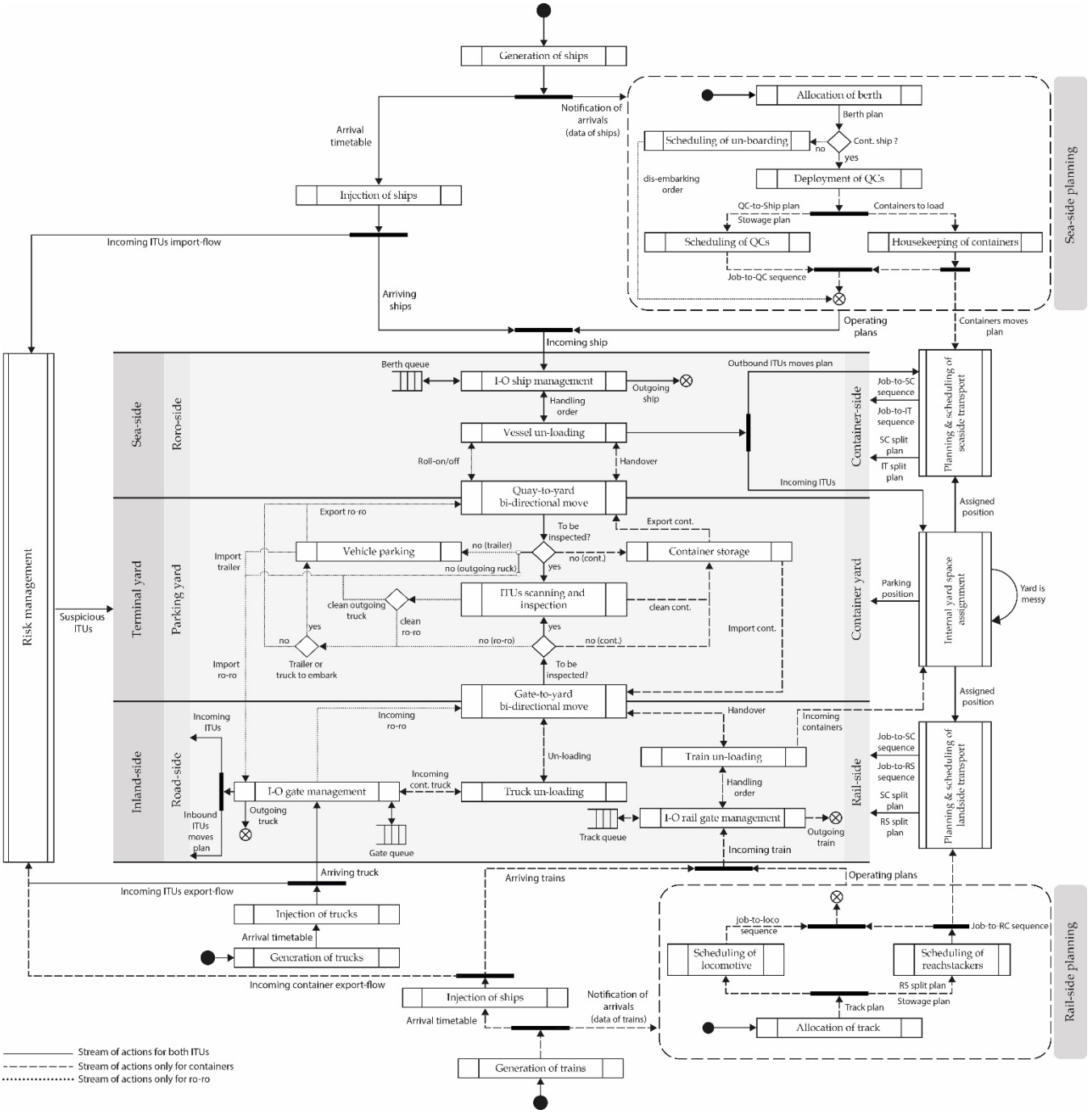

Figure 3. Global simulation model

4.2. Global design of the model

In Figure 3, the global activity diagram highlighting the dependencies among the pillars of our simulation model is exposed in a formalized form (UML specifications). This simplified representation is an aggregation of the elaborated activity diagrams. Three pools of activities are distinguished, namely truck activity, train activity, and ship activity. These activities hold the model and are a blend of decisions and operations albeit each operation can be ruled by a couple of decisions.

For trains and ships, the planning starts before their arrival at the terminal. First, the needs are determined in order to assign suitable resources and an eligible amount of equipment. For container ships, the yard housekeeping is carried out to move containers to blocks nearer to the berthing place of their respective outgoing ships so that later unloading operations will be speeded up. Once the train or ship reaches its reserved position, the handling starts. The import ITUs are unloaded, then transported to their position inside the internal yard. Conversely, export ITUs are brought from the yard to be loaded on their respective outgoing

mode. When the handling is over, the train or ship releases its place and leaves the terminal.

As concerns trucks, there is no pre-planning, decisions about resource and equipment allocation are executed once they arrive at the terminal. Container trucks are oriented toward their handover area where they are served. Once the receipt-delivery operation is achieved, the truck leaves the terminal. The Ro-Ro trucks undergo a different process depending on their type. In the case of trailers, they are moved and parked by their own tractor inside the yard and when loading, internal tractors come and grasp the trailers to drive them onto the ship. Oppositely, during the unloading, internal tractors remove trailers from the ship and park them in the yard to be retrieved later by external tractors. As for lorries, they enter and drive off ships on their own, and they are lined up in a dedicated waiting area before loading. More details were given in sub-sections 3.2 and 3.3.

5. Step III: Conception

So far, only a mesoscopic description of individual behaviors and settings has been given. In this step, the idea is to put emphasis on the conceptualization of the lower level of the system to develop a microscopic specification of our simulation model. The first sub-step is modeling the system entities and their individualities. Then, the processes are enriched with a full definition of inputs-outputs (i.e., data), and triggering and un-blocking events (i.e., messages, conditions, constraints, etc.). Meanwhile, based on functional and organizational decomposition, the system classification is carried out with the aim of reducing the system's complexity. At the end of this step, the global design of the model is refined and enriched with the improving artifacts so as to end with the conceptual model, i.e., the simplified representation to implement (linking to the next step). This updated representation outlines the way in which the considered part of reality will be reproduced.

5.1. System entities

In this approach, an entity is the basic modeling unit of the considered components of the studied system; each entity is characterized by its own independent way of existence that can be either active, reactive, or passive. Active entities are modeled as agents having goal-oriented behaviors to be able to act and interact with the environment and other entities. Reactive entities are those having stimulus-response behaviors to constantly change their internal state (physical condition) as a consequence of actions done by certain agents (e.g., moving). These entities are modeled as dynamic and immobile objects without specific goals. As regards passive entities, they are inactive objects equipped only with manipulation-based methods, principally used to represent data structures. Furthermore, the mechanism of communication, stimulus, and other ways of interaction among entities or activities are specified by messages and events.

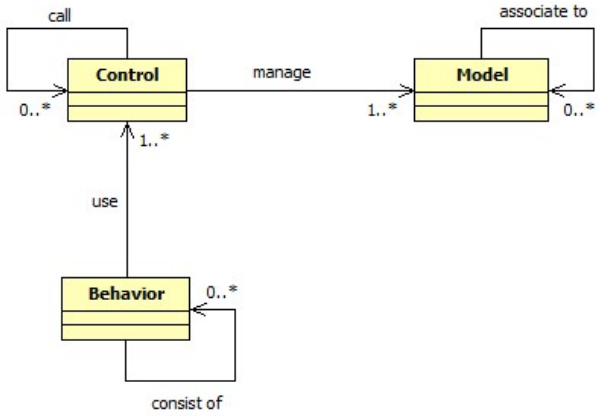

Figure 4. Model-Control-Behavior (MCB) Meta-Model

To build agents, we adopted, with a few changes, the "Entity-Control-Boundary" (ECB) design pattern (Bruegge and Dutoit 2009), for the purpose of structuration of agents and facilitating their implementation (see Figure 4). The adapted form of ECB is as follows:

- Model (entity): its original name is "entity" but here it is renamed to "model" to avoid possible confusion with the definition of entity in this paper. It is used to represent operating plans, queues, lists, terminal infrastructural resources, and database tables that are manipulated by the agent. In short, this stereotype symbolizes data and can be only associated to control classes or other model classes.
- Control: includes the logic of the agent and the treatment methods for Model classes. Indeed, a behavior is a series of actions executed to manipulate entities according to the logic of the agent and to change its internal state. Thereby, the Control class is the bridge that connects Behavior to the Model. Control classes can be associated with all stereotypes, however, a control class of an agent cannot be linked to that of another agent, because interactions between agents are only done through behaviors.
- Behavior: the term "Boundary" is replaced by "Behavior" since the interface of an agent is its behaviors. Behavior classes are used to feed the methods implemented in the Control classes with parameter values, to define roles and states of agents, and obviously to communicate. In addition, a behavior can be composed of sub-behaviors. Basically, agents possess three behaviors: listening, sending, and standing by; for reception and transmission of messages, and to wait for upcoming jobs when they have achieved their current workload, respectively.

5.2. System classification

Seeing that the studied systems are large-scale, distributed, and well-structured systems, the divide and conquer principle is called to deal with this high complexity by the way of splitting the whole into smaller and manageable sub-models or systems. Each sub-model is constructed of homogeneous components in order to play distinct roles and to be different from the others as well. All these sub-models are then plugged together using well-defined connections to act as a coherent unit to achieve the main goal for which the whole system is designed.

From the above description of maritime terminals, these basic roles can be distinguished: operating (processes), planning (decisions), generating (ITUs and transport means), displaying (key performance indicators), supporting (facilities), and executing (equipment and labor). Therefore, the global model is split into five main sub-models (sub-models can be also composed of sub-parts and so on):

- Agent sub-model: this represents equipment and labor. Agents inside this subsystem are at the beck and call of the controlling sub-model. There are four types of representative agents modeled as abstract classes: labor, handling equipment, transport equipment, and transport means. This means that these agents are not instantiable, but their properties and behaviors are inheritable. The instantiable agents are quay crane, straddle carrier, reachstacker, internal tractor agent, maneuvering agent, maintenance agent, train agent (composed of locomotive agent and wagon agent), ship agent, and truck agent (composed of external tractor agent and trailer agent). Handling and transport equipment agents have a behavior named "handling" to perform their tasks; also, labor is equipped with an additional behavior named "maneuvering", but the transport means agent does not possess any further behaviors. All of these agents have the behavior "Moving" in common.
- Object sub-model: this represents terminal facilities. Objects are unsociable entities without any awareness of their environment, yet they could be reactive. The philosophy behind the organization of objects' classes is quite similar to that of agents. The objects are mainly derived from either the abstract class "Operating Zone" (seaside, railside, roadside, and internal yard), "Resource" (berth, path, linkspan, slot (truck handling position, handover position, parking spot, and parking position), gate, weighbridge, and scanner), or purely "Area" (customs office and cell). An Operating Zone is composed of Resource and/or Area, albeit being itself an Area. In addition, an area is a set of interrelated cells (i.e., a grid), so the basic element of any area, reciprocally any resource, is a Cell object. This class is equipped with a "finite-state machine," which can be seen as a behavior. The Cell objects are able to measure constantly the degradation caused by operations (moving, dropping off containers, storing, parking, etc.) on the basis of the applied force or the degradation rate over time. When the Cell exceeds its endurance threshold (the physical condition index is under the minimum value), it shifts to a Failure state; therein an exception is thrown to inform the controlling sub-model. The cell returns back to its original state once the intervention of the maintenance agent is over.
- Controlling sub-model: this is the brain of the system seen as the more complicated and smarter sub-model since it imitates the terminal planner's reasoning as well as transport service providers. This workload is shared between these three agents: planner agent, logistics provider agent (LPA), and customs agent. As the major actor in our system, the planner agent is aware of all necessary information to work out the operating plans and decisions to be sent to representative agents. The role of the LPA revolves around the generation and synchronization of the physical flows. The LPA puts in motion transport means and informs the planner about the arrival dates of ships and trains a few days ahead for the purpose of tactical planning and the trucks arriving on the same working day. As regards the customs agent, its role is to find suspicious ITUs and to check whether they are fraudulent or not.
- Operating sub-model: as indicated by its name, this simulates the terminal's operations, where evolve facilities, equipment, and labor, and provides operation outcomes (see Figure 3).
- Dashboard sub-model: this gives a visual display of KPIs under various forms: time plots, bar charts, etc. On the dashboard, each terminal sub-system has its own KPIs that are classified into four classes: (1) utilization: describes the utilization rate of handling equipment and resources; (2) service-time: shows distance traveled and working times of handling equipment, service times at gates, and dwell times of freights and transportation modes; (3) environment: concerns greenhouse gas emission and energy consumption; (4) cost: gives the handling cost per container or trailer at each sub-system.
- User interface: where the end user tunes up system settings.

There are two ways of communication between system entities: messages as direct communication and events as indirect communication. Indeed, with the first method the sender knows exactly who the receivers are, whereas the trigger ignores who is listening to the event. Both of them are classified into four classes, as can be seen in Table 1.

Table 1. Classification of messages and events

	Classes	Description
Messages	Request	messages sent to ask the planner agent for a service or a resource.
	Reply	responses from the planner agent to the agents making a request.
	Order	orders sent by the planner agent containing operating plans to representative agents.
	Inform	to send notifications, information or data.
Events	Triggering	start running a process or an action.
	Unblocking	unlock the execution of a process.
	Exception	stop the execution of a process when a critical situation happens.
	Error	stop the simulation when a strong constraint is broken.

6. Step IV: Implementation

This step is a proof-of-concept step which concerns firstly the coding, debugging, and running of the model using AnyLogic simulation software, and secondly, validation of the simulation model to prove its ability to reflect the expected behavior. The outcome of this step is a trustful simulation model that can be used as an accurate mirror of our studied system to evaluate its performance vis-à-vis given actions, thereupon getting valuable feedback. However, we should always keep in mind the well-known quote of George Box: "*all models are wrong, but some are useful.*"

6.1. Coding and debugging

The conceptual model was implemented part by part with the help of ready-to-use AnyLogic libraries. The tools of AnyLogic used in the implementation are: Agent Library to create system agents and their behaviors; Java classes and Interfaces as well as Space Markup Elements to set up terminal facilities (object sub-model) and Control and Model classes; Process Modeling Library, Rail Library, and Road Library to represent the model activities.

To verify that the conceptual model was properly implemented, the debugging was conducted in three ways: log files, AnyLogic debugger, and a 3D animation window. The log files and AnyLogic debugger traced the execution of the model while running in order to detect any dysfunctions. A 3D animation window was used to check that the operations were correctly executed and agents behaved as expected.

6.2. Validation

As regards the validation sub-step, this can be done either by comparing the key values collected from the simulation model with those observed in the studied systems, or by the subject matter's experts. Seen that Norvik seaport terminals (see Figure 1) are new platforms that have recently opened their doors, unfortunately, observed data are not available at the moment. However, for container terminals, there is a universal key indicator to check whether containers are handled in reasonable time or not, i.e., the average handling time per container. Actually, the average handling time per container is at most three minutes (Abourraja et al. 2018), which includes one minute for the pick-up, another one for the drop-off, in addition to the equipment moving time, which could reach one minute. In our model, this indicator was about 3 minutes per container.

The designed models had been examined by experts (terminal planners) to evaluate their validity and on-the-spot visits were also arranged. Terminal planners illustrated how each activity is managed and executed. Then, several differences in our model vis-à-vis the reality were noticed:

- The entry gates for trucks are different from the exit ones; in addition, lorries and trailers enter the terminal through different gates: in our model, we assumed that trucks enter and leave the terminal via the same gate.
- Lorries and trailers are parked in different spaces: we assumed that both shared the same parking spaces.
- There is no explicit synchronization between straddle carriers and reachstackers in the rail yard: in reality first the straddle carriers move import containers to rail buffers, afterward the reachstackers start handling on the trains, and when trains leave the terminal, the straddle carriers come again to move the export containers to the internal yard. In our first assumptions, all of these operations were done simultaneously.

To be certain about the models, especially the operating and controlling sub-models, they had been discussed with terminal planners. Two main comments were made:

- The parking processes of trailers and lorries should be separated: since trailers and lorries do not undergo the same processes (see Section 4.2), a parking process named "lining-up" for lorries was added. In fact, lorries are parked in lines whilst trailers in slots.
- The human factor is missing: in our model drivers and equipment are considered as a single agent. The reason behind that was exposed as modeling human behaviors is not easy and time consuming, while using probabilities, like the majority of other research, was a better choice.

The designed models and retained insights were adapted to these observations and comments.

7. Discussion and conclusion

This paper provided a detailed approach for designing simulation models for Ro-Ro and container terminals. As with other existing approaches, it was staged into four steps with multiple sub-activities of modeling and refinement of the simplified representation in order to end with a consistent and relevant design for the studied systems. But nonetheless, it focused only on particular types of complex systems, which was not the case in the proposed approaches by Garro and Russo (2010) and Fortino and Russo (2012), etc. Moreover, our approach lacked time-saving tools or techniques that can help modelers in their duty, like generation of source code from conceptual models in the implementation step or the automatization of passage between steps; for example, the aggregation of local activity diagrams to form the global model of the system. The first point could be approximately managed through some modeling platforms such as StarUML, but they are mainly interested in generating source code for software and databases. Automatic code generation for simulation is addressed in Garro and Russo (2010) and Fortino and Russo (2012). The second point, known as model transformation, is discussed in Jouault et al. (2008).

Despite these lacks, some advantages of the approach are worth noting. The adapted form of the ECB pattern, the MCB pattern for "Model-Controller-Behavior," helped in identifying and distinguishing agent behaviors as well as in agent implementation. The given guidelines and criteria for classification can lead to a good splitting of the complex systems into a set of smaller and manageable sub-models composed of homogenous entities and components. Most notably, as far as we know, the paper on hand is the first study that integrates Ro-Ro and container terminals in a single simulation model.

The next step now is to investigate the performance of the Norvik port terminals and to perform a sensitivity analysis of the system parameters.

Acknowledgement

This article is based upon work done under the ELISA project (Energy effective Logistics and Infrastructure Systems Assessment for Cargo Ports), financed by the Swedish Energy Agency. We thank all stakeholders, particularly from Ports of Stockholm for their inputs.

References

Abourraja, M. N., Oudani, M., Samiri, M. Y., Boudebous, D., El Fazziki, A., Najib, M., Bouain, A., and Rouky, N. (2017). A Multi-Agent Based Simulation Model for Rail–Rail Transshipment: An Engineering Approach for Gantry Crane Scheduling. IEEE Access 5: 13142–56.

Abourraja, M. N., Oudani, M., Samiri, M. Y., Boukachour, J., El Fazziki, A., Bouain, A., and Najib, M. (2018). An Improving Agent-Based Engineering Strategy for Minimizing Unproductive Situations of Cranes in a Rail–Rail Transshipment Yard. SIMULATION 94 (8): 681–705.

Abourraja, M. N., Benantar, A., Rouky, N., Boudebous, D., Boukachour, J., and Duvallet, C. (2019). Towards a Simulation-Based Decision Support Tool for Container Terminal Layout Design. The 21th Int. Conf. on Harbour, Maritime & Multimodal Logistics Modelling and Simulation, Lisbon, Portugal.

Benantar, A., Abourraja, M. N., Boukachour, J., Boudebous, D., and Duvallet. C. (2020). On the Integration of Container Availability Constraints into Daily Drayage Operations Arising in France: Modelling and Optimization. Transportation Research Part E: Logistics and Transportation Review 140: 101969.

Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., and Mylopoulos. J. (2004). Tropos: An Agent-Oriented Software Development Methodology. Autonomous Agents and Multi-Agent Systems 8 (3): 203–36.

Bruegge, B., and Dutoit. A. H. (2009). Object–Oriented Software Engineering. Using UML, Patterns, and Java. Learning 5 (6): 7.

Cockburn, A. (1998). Basic Use Case Template. Humans and Technology, Technical Report 96.

Cockburn. A. (2000). Writing Effective Use Cases. Addison-Wesley Professional.

Giancarlo, F., and Russo, W. (2012). ELDAMeth: An Agent-Oriented Methodology for Simulation-Based Prototyping of Distributed Agent Systems. Information and Software Technology 54 (6): 608–24.

Garro, A., and Russo, W. (2010). EasyABMS: A Domain-Expert Oriented Methodology for Agent-Based Modeling and Simulation. Simul. Model. Pract. Theory 18 (10): 1453–67.

Günther, H. O., and Kim, K. H. (2005). Logistics Control Issues of Container Terminals and Automated Transportation Systems. Günther, H.-O., Kim, KH, Container Terminals and Automated Transport Systems.

Kemme, N. (2013). Container-Terminal Logistics. In Design and Operation of Automated Container Storage Systems, 9–52. Springer.

Kruchten, P. (2004). The Rational Unified Process: An Introduction. Addison-Wesley Professional.

Kubera, Y., Mathieu, P., and Picault, S. (2011). IODA: An Interaction-Oriented Approach for Multi-Agent Based Simulations. Autonomous Agents and Multi-Agent Systems 23 (3): 303–43.

Padgham, L., and Winikoff, M. (2002). Prometheus: A Methodology for Developing Intelligent Agents. In International Workshop on Agent-Oriented Software Engineering, 174–85. Springer.

Rouky, N., Abourraja, M. N., Boukachour, J.,

Boudebous, D., Alaoui, A., and Khoukhi, F. (2019). Simulation Optimization Based Ant Colony Algorithm for the Uncertain Quay Crane Scheduling Problem. International Journal of Industrial Engineering Computations 10 (1): 111–32.

Sindre, G., and Opdahl, A. L. (2001). Templates for Misuse Case Description. In Proceedings of the 7th International Workshop on Requirements Engineering, Foundation for Software Quality (REFSQ'2001), Switzerland. Citeseer.

Stahlbock, R., and Voß, S. (2008). Operations Research at Container Terminals: A Literature Update. Spectr. 30 (1): 1–52.

Wooldridge, M., Jennings, N.R., and Kinny. D. (2000). The Gaia Methodology for Agent-Oriented Analysis and Design. Autonomous Agents and Multi-Agent Systems 3 (3): 285–312.