MicroPort: A general simulation platform for seaport container terminals

Zhuo Sun a,⁎, Loo Hay Lee b, Ek Peng Chew b, Kok Choon Tan c

a Centre for Maritime Studies, National University of Singapore, 12 Prince George’s Park, Singapore 118411, Singapore
b Department of Industrial and Systems Engineering, National University of Singapore, 10 Kent Ridge Crescent, Singapore 119260, Singapore
c Department of Decision Sciences, NUS Business School, National University of Singapore, 15 Kent Ridge Drive, Singapore 119245, Singapore

⁎ Corresponding author. Tel.: +65 65168672; fax: +65 67756762.
E-mail address: mixwind@gmail.com (Z. Sun).

Available online 19 September 2011

A R T I C L E   I N F O

Article history:
Available online 19 September 2011

Keywords:
Container terminal
Port operations
Discrete event simulation
Multi-agent system

A B S T R A C T

Seaport container terminals are essential nodes in sea cargo transportation networks. As such, the operational efficiency of container terminals in handling containers passing through them plays a critical role in a globalized world economy. Many models and algorithms have been developed to address various decision problems in container terminals to help improve operational efficiency. These decision support tools are usually used separately for specific purposes. However, the problems they are trying to tackle are often interrelated. Therefore, in this regard, an evaluation tool which can capture as many operational conditions as possible for different decision problems is necessary. This paper introduces a general simulation platform, named MicroPort, which aims to provide an integrated and flexible modeling system for evaluating the operational capability and efficiency of different designs of seaport container terminals. The software structure of MicroPort comprises three programming layers: (1) the Functions layer; (2) the Applications layer; and (3) the Extensions layer. Different layers are bound by Application Programming Interfaces (APIs). Basic functions built in the Functions layer support the Applications layer in which major operation processes can be modeled by an agent-based method. External modules and decision support tools in the Extensions layer then use APIs to adjust the system to produce suitable simulation models for specific purposes.

© 2011 Elsevier Ltd. All rights reserved.

1. Introduction

Container terminals play a critical role in a globalized world economy. At a container terminal, the operator is responsible for handling containers from sea to shore and vice versa in accordance with stowage instructions from shipping lines (their customers). In a container terminal (Fig. 1), vessels from different shipping lines arrive based on some predetermined schedules. Ship-to-shore cranes, or simply quay cranes, unload containers from vessels to yard trucks or load containers from yard trucks to vessels. Yard trucks travel between quay cranes and yard cranes to transport containers. Yard cranes straddle in the yard and handle containers between yard trucks and container stacks.

Much co-ordination is needed between pieces of equipment to ensure that a seamless flow of containers can be achieved. This requires varied planning which ranges from long-term strategies to real-time operational decisions. Due to the size and complexity of the problem, usually port operators employ different teams of people to take charge of different processes. In each of the processes, different levels of planning are required. For example, in berth planning, the port operator needs to determine ahead of time the assignment of vessels to berths, and the allocation of quay cranes to serve these vessels based on the vessels’ scheduled arrival times. However, actual arrival times of vessels tend to deviate from their scheduled arrival times due to delays or other unforeseen circumstances. Hence, real-time decisions to reassign vessels to berths as well as quay cranes to these vessels are often necessary. When making such decisions, the port operator aims at maximizing port efficiency, which includes minimizing the vessel turnaround time, maximizing the quay crane productivity, and maximizing the throughput of container terminals. However, in practice, this objective might not be achievable. This is because these decisions are often made hierarchically and locally. Moreover, some decisions are supported by sophisticated decision support tools, while others are made on an ad hoc basis. Hence, one cannot be sure if the system had performed efficiently as a whole because there are often significant interactions among these decision processes, such as, for example, between berth allocation and yard management. The problem is compounded by the fact that there are many uncertainties existing in the system, e.g., vessel delays, equipment breakdowns, etc.

To effectively evaluate these decision processes, we need an evaluation tool which can effectively capture the nature of the complex and interrelated systems for incorporating sophisticated decision support tools (which can be commercial software or
existing decision support tools used by port operators). From the literature, although there are many simulation models for container terminals that have been built, these are often too rigid and thus incapable of adapting to different terminal configurations. In this paper, we propose a general simulation platform, named MicroPort, which can provide an integrated and flexible modeling system for evaluating the operational capability and efficiency of different designs of seaport container terminals. The software structure of MicroPort comprises three programming layers where lower layers support higher layers with Application Programming Interfaces (APIs). Basic functions are built in the lowest layer. Major operation processes are modeled as an improved multi-agent system in the middle layer. Finally, the highest layer contains external programs which can adjust the system to produce suitable simulation models through APIs.

Organization of the rest of the paper is as follows. A review of previous related works is given in Section 2. The software design of the proposed simulation platform is then presented in Section 3. The simulation software structure is described in Section 4 followed by verification and validation of the platform in Section 5. Case studies are given in Section 6. Conclusions are drawn in Section 7.

2. Literature review

Recently, due to the rapid development in computer technology, simulation has been extensively used to study container terminal operations. We first go through some simulation models focusing on specific operation processes in container terminals. Then some integrated simulation models will be discussed which are followed by several multi-agent systems. Finally, a comparison table is given.

Numerous studies created simulation models for container terminals and used them as subsystems for testing, evaluating and forecasting their decision making algorithms and strategies. Gambardella et al. [1] built a decision support system focused on the resource allocation problem and used a terminal simulation model as a test bed for checking the validity and robustness of the system. Kia et al. [2] compared two different operational systems (current and proposed) statistically via a simulation model and proposed an operational method to reduce terminal congestion and increase terminal capacity. Nam et al. [3] used a simulation model with four scenarios to examine the optimal size for the Gamman container terminal in terms of berths and quay cranes. Demirci [4] used simulation experiments to determine the most critical bottleneck processes in the port system, and an investment strategy was developed to balance the load in the port. Lee et al. [5] developed a simulation model to evaluate port operations in a supply-chain network. Sgouridis et al. [6] created a simulation model to study the handling of incoming containers transported by trucks and used the model for short to long term planning as well as for process improvement. Liu et al. [7] developed simulation models to study the impact of automation and terminal layouts on terminal performance. In particular, two different terminals with different yard configurations are considered. Parola and Sciomachen [8] tried to simulate the logistics chain of the northwestern Italian port system as a whole. Possible future growth of container flow was evaluated. Duinkerken et al. [9] developed a simulation model equipped with a rule-based control system to compare two container transportation systems: multi-trailer automated guided vehicles and automated lifting vehicles. Lee et al. [10,11] developed simulation models to investigate how different vehicles and different yard layouts can affect the efficiency of the port operations. They further build a program named Automated Layout Generation to generate different simulation models. Ha et al. [12] presented a 3-D real-time-visualization simulation model which depicts very detailed behaviors of equipment in container terminals. This model is useful for assessment of the performance for new equipment. Hadjiconstantinou and Ma [13] developed a decision support system to optimize yard operations by considering all container flows through the yard and used a simulation model for validation. Zeng and Yang [14] developed a simulation model for scheduling loading operations in container terminals and then embedded this model in an optimization method to determine the optimal scheduling scheme.

Although the above simulation models can well represent certain parts or processes in a container terminal, for a more realistic representation of the actual operations, it is necessary to treat inter-related subsystems as a whole. Only few studies considered integrative simulation models for container terminals. These models covered most aspects of container terminal operations without losing certain details. Nevins et al. [15,16] developed a seaport simulation model that computes throughput capability and determines resource utilization at a high level of details. The simulation model allows for multiple cargo types as well as
multiple ship types. Shabayek and Yeung [17] created a simulation model to simulate Kwai Chung container terminals to study the extent by which a simulation model could predict the actual container terminal operations with a high level of accuracy. Hartmann [18] introduced an approach for generating scenarios for sea port container terminals. The scenarios can be used as input data for simulation models. Furthermore, they can be employed as test data for algorithms solving optimization problems in container terminal logistics such as berth planning and crane scheduling. Bielli et al. [19] used an object oriented design for developing a simulator for container terminals. Every piece of equipment, queue, or area is implemented as an object and communications among objects are implemented as messages. Ottjes et al. [20] introduced a generic simulation model structure for the design and evaluation of multi-terminal systems. This model is constructed by combining three basic functions: transport, transfer, and stacking. Petering [21] developed a comprehensive simulation model to address issues in terminal design, storage and retrieval location and yard crane control. None of these studies were able to provide flexible integrated solutions for different container terminals. Hence, in this study, we aim to introduce a general platform to generate simulation models for various container terminals.

Normally simulation models for container terminals use a centralized structure which implements a control center controlling every event and process. However, in a typical container terminal, the equipment might be manually driven by human drivers and might interact with each other to form a complex system. The complexities of such systems cannot be easily modeled by a centralized monotonot logic but rather by decentralized autonomous agents. Several studies developed simulation models using a multi-agent approach. Rebollo et al. [22] developed a multi-agent system to simulate the port container terminal management. Henseney et al. [23–27] employed a multi-agent based simulation approach for the evaluation of container terminal management operations. The approach aims to plan and coordinate the processes within the terminal by mapping the terminal’s objects and resources. Franz et al. [28] presented a multi-agent based simulator to simulate market-mechanisms in container terminal management.

The models covered in this literature review are categorized in Table 1. Commercial software products are compared in Table 2 from the external functionalities’ perspective as their internal mechanism is always protected. The simulation platform proposed in this paper differs from the other simulation models available mainly on the software structure. Based on the hybrid modeling structure, different simulation models can be generated to fit different needs. Therefore its external functionalities are diverse even compared to commercial software.

### Table 1

Overview of simulation research for container terminals.

<table>
<thead>
<tr>
<th>Simulation target</th>
<th>Simulation method</th>
<th>Related research</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminal subsystem</td>
<td>Discrete event simulation</td>
<td>[1–14]</td>
</tr>
<tr>
<td>Whole terminal</td>
<td>Discrete event simulation</td>
<td>[15–21]</td>
</tr>
<tr>
<td>Multi-agent system</td>
<td>MicroPort (this paper)</td>
<td></td>
</tr>
<tr>
<td>Hybrid modeling system</td>
<td>MicroPort GIS + 3D + Dynamic UI</td>
<td>C/C++, Lua</td>
</tr>
</tbody>
</table>

### Table 2

Comparison of functionality in simulation software and MicroPort.

<table>
<thead>
<tr>
<th>Simulation software</th>
<th>User interface</th>
<th>Programming interface</th>
<th>Container terminal related tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anylogic</td>
<td>2D + 3D + Dynamic UI</td>
<td>Java</td>
<td>None</td>
</tr>
<tr>
<td>Arena</td>
<td>2D + 3D</td>
<td>Visual Basic</td>
<td>None</td>
</tr>
<tr>
<td>Automod</td>
<td>2D + 3D</td>
<td>Automod</td>
<td>None</td>
</tr>
<tr>
<td>Plant simulation</td>
<td>2D + 3D</td>
<td>SimTalk</td>
<td>None</td>
</tr>
<tr>
<td>ExtendSim</td>
<td>2D + 3D</td>
<td>Modl</td>
<td>None</td>
</tr>
<tr>
<td>Flexsim</td>
<td>2D + 3D</td>
<td>C++</td>
<td>Flexsim CT library</td>
</tr>
<tr>
<td>ProModel</td>
<td>2D + 3D</td>
<td>ProModel</td>
<td>None</td>
</tr>
<tr>
<td>Witness</td>
<td>2D + 3D</td>
<td>Witness</td>
<td>None</td>
</tr>
<tr>
<td>Delmia Quest</td>
<td>2D + 3D</td>
<td>C++</td>
<td>None</td>
</tr>
<tr>
<td>Non-commercial tools reviewed by this paper</td>
<td>None</td>
<td>Various</td>
<td>Dedicated</td>
</tr>
<tr>
<td>MicroPort</td>
<td>GIS + 3D + Dynamic UI</td>
<td>C/C++, Lua</td>
<td>Generalized port operation modules</td>
</tr>
</tbody>
</table>

### 3. Software structure of MicroPort

The software structure of MicroPort comprises three programming layers; namely: (1) Functions layer; (2) Applications layer; and (3) Extensions layer (Fig. 2). In the Functions layer, many basic functions are implemented to support higher layers. In the Applications layer, an improved multi-agent system has been built to represent interactions among various types of equipment and major decision processes. In the Extensions layer, user modules can be plugged in for implementing simulation initiations, graphical user interfaces, dynamic graphic outputs, etc. These layers are further elaborated in the following subsections.

#### 3.1. Functions layer

As the base layer of MicroPort, the Functions layer should be fast, versatile and flexible. After comparing a number of different simulation software, MicroCity [29] was chosen to be the backbone of MicroPort because of the following reasons. Firstly, MicroCity has an open architecture which allows it to be easily extended to fulfill various needs. Secondly, there are many functions already provided by MicroCity (such as Geographic Information System, Network solver, Mixed Integer Programming solver, 3-D real-time render engine, etc.) which are not equipped in other software. Finally, MicroCity guarantees very high running speed with the use of C/C++ as its low-level programming language and Lua [30] as its higher level scripting language. MicroCity does not directly provide simulation related functions in its lower layer. Although these functions can be implemented in higher layers without much difficulty, using lower level language to implement these functions will provide big advantages in terms of performance.

In the Functions layer, following MicroCity, basic functions are implemented in C/C++ and registered as Lua functions which can be executed from higher layers. There are two series of functions built in the Functions layer to implement basic simulation functionalities. The first consists of random number generators, and the second comprises discrete-event schedulers. A Mersenne Twister random number generator (Matsumoto and Nishimura [31]) coupled with a transformation function is implemented to facilitate the generation of random numbers for popular probability distributions.

For discrete-event schedulers, the conventional implementation is simple. The main body of the program maintains a priority queue in which events are sorted in ascending order of their start times.
When an event is popped from the queue, the associated subroutine will be triggered. However, both conventional procedure-oriented and object-oriented programming languages are not able to preserve the running environment from one process to another if the two processes are related in data or logic. On the other hand, in modern dynamic programming languages, like Lua, we can easily suspend and resume the current program state through cooperative threads (Jouvin [32]). Therefore, this study binds Lua co-routines (generalized subroutines to allow multiple entry points for suspending and resuming execution at certain locations) to events to preserve running contexts. If there are no relationships among events, the associated co-routines will act as subroutines. If multiple events belong to the same entity or operation process, a single co-routine will be used. This co-routine will be created by an initialization event, suspended and later resumed by relay events (Fig. 3) without any extra efforts at storing data or logic. This event execution pattern in the Functions layer of MicroPort gives flexibility to higher layers. Serial, concurrent, or serial-concurrent mixed modeling concepts can be easily implemented in higher layers.

3.2. Applications layer

The Applications layer of MicroPort aims to represent the major operation processes of container terminals. There are two major types of control systems in current container terminals; namely, centralized systems and distributed systems. A centralized system has a control center which makes all decisions and controls the movement and operations of all equipment. Centralized systems are currently widely used in container terminals and simulation models. However, it is critical to observe that a container terminal is a highly complex inter-related system. Many interactions between individual equipment are difficult to be directly or exactly modeled in a centralized system. Thus, there is a boom recently in the development of multi-agent systems in which various equipment types and decision making processes are represented by specific software agents.

By using the proposed discrete-event schedulers in the Functions layer, an improved multi-agent system has been implemented in the Applications layer. As illustrated in Fig. 4, the proposed multi-agent system has a main thread and many agent co-routines. The main thread is able to switch the system between serial mode and concurrent mode. In the serial mode, the system halts all agents and executes a sequence of central controlling procedures. These procedures make overall decisions based on the current system state. When the system goes into the concurrent mode, distributed agent co-routines will be activated and they interact with each other in the context of overall decision making results. For instance, there are dozens of yard trucks traveling in the yard and interacting with other piece of equipment. The jobs for each yard truck are scheduled by the yard truck scheduler in the control center of the terminal. To model this subsystem in MicroPort, every yard truck can be represented by an agent co-routine and the yard truck scheduler by a central controlling procedure. The yard truck scheduler will be regularly triggered by events to pause all agent co-routines for making plans for yard trucks. Then the simulation will resume and all of the agent co-routines will be activated simultaneously to continue the yard truck operations. Continuations provided by the lower layers of MicroPort can make individual agent co-routines proceed smoothly without data or logic breaks.
As the whole Applications layer is constructed with the dynamic programming language, Lua, implementing any existing modeling concept for further development can be easily done.

3.3. Extensions layer

The Extensions layer is created for extending MicroPort on case-related functionalities. It is based on Application Programming Interfaces (APIs) provided by the Functions and Applications layers in MicroPort (Fig. 5). Modules in the Extensions layer are loosely organized by Lua and can be plugged in at any time. These modules include Graphic User Interfaces (GUIs), Graphic Output, Logging, Terminal Layout Generation, Replaceable User Algorithms, etc. Developers can choose C/C++ to build modules for high performance or Lua for high flexibility. Other software can also use C/C++ or Lua to communicate with MicroPort. When using external software along with MicroPort, it is best to use MicroPort as the host program to control the whole communication process. Data between MicroPort and any external software can be exchanged through APIs, files and/or system memory.

Modules in the Extensions layer can be packed with data sets as projects. Each project has its own data sets and modules for modeling a particular container terminal. MicroPort provides a default simulation structure which can be used as a template for new projects.

4. The default simulation structure

MicroPort’s software layers provide a powerful and flexible simulation modeling environment. This section introduces the default simulation structure which is mainly modeled in the Applications layer and includes two major parts: decision processes and equipment interactions. Decision processes are modeled as serial procedures in MicroPort. Pieces of equipment are modeled as agents and their interactions are modeled as concurrent events of agents. Fig. 6 shows the overall simulation flow. The simulation repeatedly generates vessel arriving events. These events then trigger a series of decision processes. These processes make long-term, short-term and real-time decisions for various types of terminal equipment which, in turn, set job schedules for the terminal equipment. The equipment takes their job schedules as prior rules and interacts with each other to move containers around the terminal. They will in turn trigger decision processes when they reach certain states. The following subsections will describe the decision processes and equipment interactions respectively.

4.1. Decision processes

There are several default decision processes in MicroPort; namely, Berth Allocation, Quay Crane Assignment, Quay Crane Scheduling, Container Location Assignment, Yard Crane Deployment, Yard Crane Dispatching, and Yard Truck Dispatching. Most of the decision problems in major container terminals are covered by these decision processes. Other models of container terminals can also be adapted by using MicroPort as a template with slight modifications. Fig. 7 shows the overall relationships among simulation and decision processes. These user specified decision algorithms are stored externally and loaded dynamically during runtime. APIs provided in the Extensions layer facilitate the loading process. The default decision procedures in MicroPort are elaborated below.

The Berth Allocation procedure and the Quay Crane Assignment procedure in MicroPort allocates limited berth space for vessels and assigns a certain linear set of quay cranes to each vessel. This procedure performs a deterministic long-term berth allocation and quay crane assignment based on long-term vessel arrival schedule provided in the initialization phase. The results are then used as templates for real-time berth allocation and quay crane assignment.

Each assigned quay crane has a task list in which containers will be discharged or loaded in a certain order. In practice, this step is quite complicated. Decisions are often made through a combination of computerized heuristic rules and human decisions. In MicroPort, the default Quay Crane Scheduling procedure simplifies this step by sequentially enumerating all discharging and loading
containers for a vessel and appending them to the task lists of quay cranes.

Every container in a quay crane’s task list should be assigned a yard storage location before handling. This assignment can be done right after the task list has been built up or just before the quay crane starts handling the container. The short-term deterministic Container Location Assignment module is executed when urgent decisions are not needed. The solution obtained from short-term assignment will then be used as a template for real-time decision making which is executed just before a quay crane starts handling a container in the quay crane’s task list.

MicroPort provides two procedures (Yard Crane Deployment and Yard Crane Dispatching) for managing movements of yard cranes. The Yard Crane Deployment procedure assesses all the yard locations of assigned quay crane tasks and deploys a certain number of yard cranes to each yard zone. Then every deployed yard crane in a zone will be responsible for handling container jobs in an area consisting of a number of consecutive slots. Yard crane deployment algorithms can have very complicated logic through external modules. However, by default, MicroPort will attempt to assign approximately equal numbers of future container jobs to every yard crane’s planning horizon. Thereafter, the Yard Crane Dispatching procedure will make short-term deterministic schedules. These schedules are used as templates to make real-time yard crane dispatching decisions.

The last and the most frequently triggered procedure in MicroPort is the Yard Truck Dispatching procedure. When it is triggered by other decision making procedures, the short-term deterministic
yard truck dispatching module will be executed. Because of the ever-changing environment of the yard, this module only considers limited number of future container jobs in the quay crane task lists and always incorporating with short-term deterministic yard crane dispatching procedure to make globally optimized decisions. When the Yard Truck Dispatching procedure is triggered by waiting or moving yard trucks, real-time decision making unit should respond immediately. These real-time decisions are based on short-term planning results and can have many detailed instructions including destinations and routing information. This procedure is not only suited for yard trucks but also can be adapted for other transportation equipment in the yard. Different user-specified algorithms can be plugged in through external modules.

4.2. Equipment Interactions

The results produced by decision processes are translated to instructions for terminal equipment. There are three major types of equipment in MicroPort: quay crane, yard crane and yard truck. The interactions among these pieces of equipment can be categorized into five types: interference between quay cranes, interference between yard cranes, interference between yard trucks, interactions between quay cranes and yard trucks, interactions between yard cranes and yard trucks. Because quay cranes and yard cranes in MiroPort are assumed to move on a single lane, they cannot move through each other. Hence the minimum safety spacing between quay cranes and yard cranes are mainly controlled by the decision processes. The minimum safety spacing for quay cranes and yard cranes can be set before the start of a simulation run. In simulation, when two quay cranes or yard cranes get too close, the corresponding decision process will be triggered to re-arrange the schedules of the two pieces of equipment to change the current moving destinations of at least one of the two. The interference between two yard trucks can cause collisions. In simulation, autonomous agents are responsible for avoiding collisions. When two yard trucks get too close, each of them will make a detour but still head to the same destinations.

As to the interactions between quay cranes and yard trucks, as well as interactions between yard cranes and yard trucks, Fig. 8 illustrates the control and data structure used in MicroPort. When a yard truck is dispatched to a quay crane for loading or discharging jobs, it will be added to the waiting queue associated with the quay crane. The quay crane will then decide which yard truck in the queue should be served first. When a yard truck is dispatched to a yard location, it will be added to the waiting queue associated with the corresponding yard slot. A yard crane waiting at the slot, or to be dispatched to this slot, will serve the yard trucks in the queue in a first-come-first-serve manner.

In multi-agent systems, deadlocks are common issues because agents compete for limited resources. In MicroPort, agents are implemented by using collaborative co-routines, which can help eliminate disorder of agent's activities. However, logically deadlocks can still happen when several pieces of equipment are in a waiting cycle. Since quay cranes' schedules are often hard to change, the key for avoiding deadlocks is through the control of yard cranes and yard trucks. To solve this problem, in real-time decision making units, MicroPort will only consider those pieces of equipment that are to appear at the location. This will ensure that the equipment that is waiting will not be waiting for another piece of equipment that may never show up.

5. Verification and validation

To ensure that various simulation models for container terminals are correctly implemented, verification is the most important process. MicroPort has several features to facilitate this process.

(1) The debuggers. Although MicroPort itself does not contain a stand-alone debugger to debug simulation models, there are many sophisticated debuggers for Lua. Generally speaking, concurrent multi-agent systems are very hard to debug. However, because the agent entities in MicroPort are implemented in collaborative co-routines, Lua debuggers can easily pause the whole simulation and trace specific agents' behaviors.
(2) The logging service. Debugging a complicated process is not always efficient at runtime. The logging service provided by MicroPort can log desired information at runtime without breaking the simulation and then perform static analyses afterwards.

(3) The graphic output. Undoubtedly, visualized data is far more understandable than plain numbers. MicroPort provides APIs which can output not only 2-D animations and charts, but also 3-D animations. Decision processes and especially interactions among pieces of equipment can be intuitively examined. Besides, the built-in GIS functions in MicroCity give more accuracy and flexibility in showing computer graphics.

After building a simulation model, this model should be calibrated in order to represent the real system as accurately as possible. MicroPort offers the following methods to facilitate the validation process.

(1) The modular input interfaces. MicroPort embeds a set of input functions to load and save simulation parameters. Users can not only calibrate the parameters through the graphic user interface, but also dynamically calibrate parameters in simulation models.

(2) The image service. MicroPort can import satellite images and aerial images and calibrate simulation models according to the imported images.

(3) The terminal evaluation utilities. There are several indicators for evaluating the efficiency of container terminals, such as vessel turnaround time, vessel waiting time, terminal throughput, berth occupancy rate, yard occupancy rate, gross crane rate, etc. MicroPort provides a set of utilities to do these calculations. Simulation models can be calibrated by comparing with a terminal’s actual indicators.

6. Case study

To test the practicability of MicroPort, we use it to investigate operational problems of a real transshipment container terminal located at Singapore. The operator wants to improve the current terminal’s productivity without great changes. We first configure MicroPort to fit for the current status of the terminal then perform several experiments on the simulation model to find out the possible solutions.

The geographical layout of this terminal is extracted from a satellite map (Fig. 9). This terminal has a 1200 m quay and a 1100 m by 300 m yard. There are 15 quay cranes, 63 yard cranes and 60 yard trucks in the terminal. The simulation model generated by MicroPort includes default operational policies for transshipment container terminals. We adjusted the parameters and conducted six independent simulation runs. The warm-up period is set to be three weeks. The data collection period in each run starts from the 4th week and lasts for 12 weeks. The statuses of the terminal resources can be immediately presented during the simulation execution (Fig. 9). This information is used to trace the activities of quay cranes, yard cranes and yard trucks and calibrate the simulation model. Table 3 shows averaged results summarized from the six complete executions. The most important result, Gross Crane Rate (GCR) which measures the quay crane productivity (lifts per hour) is identical to the realistic observations. This long-run GCR not only depends on quay cranes’ characteristics but also can be affected by other equipment and operating policies.
used in the terminal. Improvements can be made through investment on new equipment as well as adoption of new operating policies. We need to examine potential improvement scenarios in our simulation model to help the terminal operator achieve a higher GCR.

There are four sets of experiments (Figs. 10–13) performed on this model to show the sensitivity of a single parameter affecting quay crane productivity while other parameters are fixed. In each set of experiments, a common random number scheme is used. Every experiment consists of six independent simulation runs, where the data collection period in each run is also 12 weeks.

Fig. 10 shows the relationship between the number of yard trucks and quay crane productivity (GCR). As the number of yard truck increases, GCR first increases steeply then becomes stable. Similar situation also happens when increasing the number of yard cranes (Fig. 11). If a container terminal has limited resources, these experiments can help the terminal operator decide reasonable investment on equipment.

Fig. 12 depicts that the increasing of minimum safety spacing between every two yard cranes can hamper the quay crane productivity. As the minimum safety spacing between every two yard cranes increases, the quay crane productivity steadily decreases. It is a trade-off between the safety and productivity. From the terminal operator’s point of view, safety is one of the most important issues that must be guaranteed. Therefore he cannot reduce the minimum safety spacing too much to increase the productivity.

Yard physical condition is another impact factor to the productivity of the whole terminal (Fig. 13). Small yard block lengths can shorten horizontal travel distances for yard trucks and yard cranes and thereby speed quay cranes’ efficiency. But the yard width should be increased to maintain the same capacity, which may be costly in land-scarce terminals.

Based on the above four experiments, three improvement scenarios have been proposed and tested (Table 4). The main mechanisms of the simulation model are remained same. The only thing need to be changed is resource. Because of the layered framework structure that preserves generalized operation logics in the Applications layer, we just made a specific module in the Extensions layer for scenario generation and evaluation. The investment increases as the number of yard trucks and yard cranes increases. Shortening yard length can get extra productivity gain when the yard trucks and yard cranes are “saturated” in the yard. Although shortening yard length is the cheapest scenario for improving terminal productivity, to maintain yard capacity, the part of the increased width may be costly as discussed before. Hence we chose three appropriate yard lengths which can keep the yard capacity above the maximum requirement without changing width. The result shows a similar trend as previous individual experiments. The more investment the operator puts the slower increasing productivity he gets. The most economic decision is obviously scenario 1 in which the operator put less money but get more improvement comparing the others.

Table 3
Summarized simulation results.

<table>
<thead>
<tr>
<th>Indicators</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vessel turnaround time (h)</td>
<td>45.82</td>
</tr>
<tr>
<td>Quay crane productivity (GCR)</td>
<td>36.23</td>
</tr>
<tr>
<td>Yard crane productivity (GCR)</td>
<td>22.82</td>
</tr>
<tr>
<td>Yard truck productivity (GTR)</td>
<td>7.23</td>
</tr>
<tr>
<td>Maximum annual throughput (TEU)</td>
<td>4,760,520</td>
</tr>
<tr>
<td>Total quay crane utilization</td>
<td>76.39%</td>
</tr>
<tr>
<td>Total yard crane utilization</td>
<td>63.75%</td>
</tr>
<tr>
<td>Average berth utilization</td>
<td>56.38%</td>
</tr>
<tr>
<td>Average yard utilization</td>
<td>58.42%</td>
</tr>
</tbody>
</table>

Fig. 10. Quay crane GCR for various number of yard trucks.

Fig. 11. Quay crane GCR for various number of yard cranes.

Fig. 12. Quay crane GCR for various minimum yard crane spacing.

Fig. 13. Quay crane GCR for various yard length.
7. Conclusions

This study introduces a general simulation platform, named MicroPort, for container terminals. The three software layers in MicroPort provide flexibilities in developing various types of simulation models. In the Applications layer, this study proposed a generalized simulation structure which uses an improved multi-agent system to represent various operation processes and different equipment in container terminals. The system provides default decision processes covering major long-term, short-term and real-time planning problems in typical container terminals. From the case study, we can see that a particular simulation model for a real container terminal can be built based on this platform by using specified modules for actual configurations. Various analyses can be performed by using the platform to provide useful and insightful results for port operators and port planners and designers. It has been shown in the case study that the platform will greatly reduce the modeling efforts and contribute to the planning and design of integrated solutions for container terminals.

Acknowledgements

The authors would like to thank all colleagues and students who contributed to this study and all the reviewers for their valuable comments.

References