D²CTS: A DYNAMIC AND DISTRIBUTED CONTAINER TERMINAL SIMULATOR

G. Lesauvage[(a)], S. Balev[(b)], F. Guinand[(c)]

Université du Havre, LITIS EA 4108
BP 540, 76058 Le Havre, France

[(a)]gaetan.lesauvage@litislab.eu, [(b)]stefan.balev@litislab.eu, [(c)]frederic.guinand@litislab.eu

ABSTRACT
The CALAS project (Carrier Laser Tracking System) consists in a laser measure system able to localize precisely straddle carriers within a container terminal. The information given by such a tool makes an optimization possible. As members of the LITIS, our participation in this project concerns the conception and the development of a simulation platform able to reproduce both structure and dynamics of a container terminal. The software must be able to simulate dynamic events occurring during a container terminal day of work. Therefore, we proposed D²CTS, a Dynamic and Distributed Container Terminal Simulator. We will discuss in this paper the modelling process and the possibilities of such a simulator.

Keywords: container terminal, modelling, simulation, dynamics.

1. CONTEXT

With the development of trade activities which have continually increased, container has become the first mode of packaging for exchanging goods. Container terminals have been created all around the world in order to facilitate the transfer between ships and trucks or trains. The performance of these transfers has to be considered to reduce the waiting cost of the container terminal customers.

Le Havre's harbour is the biggest harbour of France in container traffic. It is located at the North West cost of France, beside the Channel, sea door between the Atlantic and the North Sea. To keep competitive, the harbour has to provide a high quality of service and unceasingly develop new technologies and processes.

To compute optimized solutions for problems such as berth allocation, vehicle routing, mission scheduling or container positioning, the location of entities present within the terminal must be known. GPS systems were traditionally used to locate the vehicles but this technology is not accurate enough regarding the terminal configuration. In fact, the distance between two lanes can be less than the precision of GPS. So more accurate technology have been developed such as differential GPS (DGPS) or laser systems which are much more accurate.

CALAS project (acronym for Carrier Laser Tracking System) concerns a laser localization system created by Laser Data Technology Terminal (LDTT). This project takes place in Le Havre's harbour in France and regroups multiple partners such as industrial companies (*Ateliers de Normandie*, EADS Astrium, Electronic Equipment and D2A), research laboratories (LMAH, LITIS) and universities (University of Le Havre, INSA of Rouen). LDTT's technology is composed by a set of laser data access points spread all over the terminal and laser sensors able to send data such as location, current task, direction, etc. to the optimization system in real time.

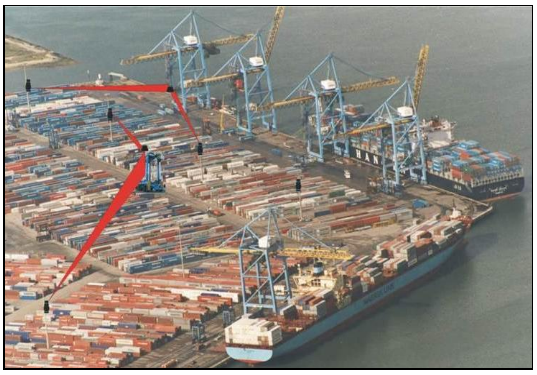


Figure 1: Laser Heads located on the *Terminal de Normandie*, Le Havre, France (source: http://www.ldtt-fr.com/).

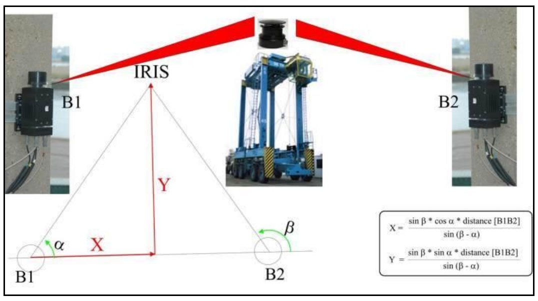

IRIS
B1
B2
Y
α
X
β
B1
B2
X = sin β * cos α * distance [B1B2] / sin (β - α)
Y = sin β * sin α * distance [B1B2] / sin (β - α)

Figure 2: LDTT's laser localization system (source: http://www.ldtt.fr.com).

D²CTS (Dynamic and Distributed Container Terminal Simulator) is our contribution to this project. It models a container terminal and all the interactions occurring in this complex system. Our purpose was to represent the terminal both in its structure and its dynamics. D²CTS aims at being coupled to the laser data system and at running optimization algorithms dealing with the data sent by the sensors and finally at sending the results to the entities through the communication system.

2. MODEL

As usual with complex systems modelling, the level of precision of the model must be considered carefully. Indeed, a useless level of detail will degrade the performance (Heidemann et al. 2001) and, on the other hand, a lack of detail will generate inaccurate data (Cavin et al. 2002).

We chose to use discrete time to be able to control the level of details in the time dimension. Moreover, to be able to model such a large scaled simulation, we used a parallel and distributed simulation architecture (Fujimoto 2000).

2.1. Container terminal organization

A container terminal is generally divided in three parts. First, the quay side where ships are loaded or unloaded. On the opposite, the land side concerns both trucks and trains. And finally in between these two areas the stocking area (yard) is used to stock temporary containers within the terminal. Each zone is composed by one or several blocks. Each block is formed by a set of lanes. And each lane is composed by a set of slots.

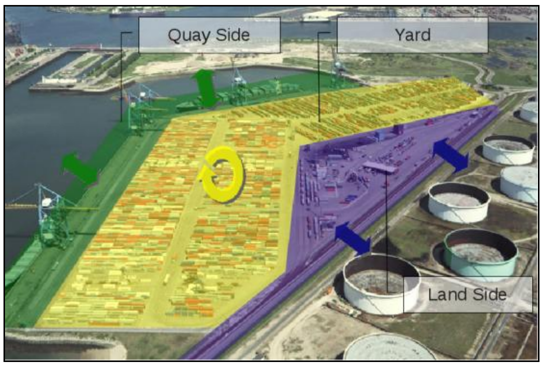


Figure 3: The three container terminal areas.

The trucks area has been modelled by lanes of only one slot able to stack only one container high if and only if a truck is parked at this location. The train areas have also been modelled by lanes of slots of one container high but those lanes can be discontinuous to let other vehicles going through. A container can be stack on a lane of this area only if a train is there too. In the yard area, a slot is generally able to contain a stack of three containers high according to the handling trucks characteristics.

2.2. Container terminal vehicles

As described in (Steenken, Voss, and Stahlbock 2004), vehicles found within a container terminal can be divided in two categories. The first one concerns the customers' vehicles such as trucks, trains or ships. The second one concerns container handling vehicles.

The vertical handling facilities are all the cranes such as quay cranes (or gantry cranes) or stocking cranes. These structures are respectively used to load or unload ships and to stock a container in the yard.

Figure 4: Quay cranes used at Le Havre's harbour (source: http://www.t-n.fr).

On the other hand, the horizontal handling category regroups vehicles able to move a container from a place to another one into the container terminal.

Some of them are passive, that is they cannot load or unload a container by their own means, they generally need a crane. Among these vehicles, the automated guided vehicles (AGV) are able to drive within the terminal thanks to an electric wire network.

On the contrary, the active handling trucks are able to lift a container by themselves. These vehicles are straddle carriers, forklift trucks or reach stackers. Forklift trucks are generally used to handle empty containers whereas reach stackers which can take a container by the side, are used to load containers on a train. Straddle carriers can lift a container from above and are very useful to move containers in the yard by driving over the lanes. They are also used to load or unload trucks or trains. Some of them are able to dynamically adapt the spreader size to any container dimensions while some of them require to be set up in the depot.

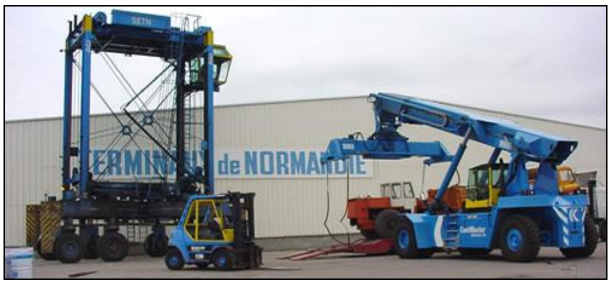


Figure 5: straddle carrier, forklift truck and a reach stacker used in the *Terminal de Normandie* (source: http://www.t-n.fr).

2.3. Container terminal road network

The blocks of the terminal are linked by a network of roads and crossroads. We can model it by a directed graph in which the nodes are the crossroads and the edges are both the roads and the lanes.

The two types of edges are handled differently, because two straddle carriers cannot cross when they are on the same lane. That is why lanes are modelled as First-In-First-Out edges (Orda and Rom 1990). Moreover, the capacity of a lane is usually limited to one vehicle at a time. So other vehicles will have to wait for the lane to be freed before using it. This characteristic has a consequent impact on the routing performance and has to be considered by the routing algorithms (Lesauvage, Balev and Guinand, 2011).

2.4. Straddle carrier activities

For the moment, this simulator focuses on the straddle carrier activities. As described above, these vehicles are autonomous and are widely used for this reason. A straddle carrier moves containers within the terminal. Those moves are called missions. We distinguish four kinds of missions:

- Incoming container missions;
- Outgoing container missions;
- Transhipment missions;
- Staying container missions.

The first category concerns trucks, trains and ships unloading. Straddle carriers drive to the pick-up locations and unload the vehicles, and then lift the container, drive to the yard to stock it. Concerning ships, they are unloaded by quay cranes which stack the containers on the quay. Then, straddle carriers come to pick-up the containers. The second category concerns trucks, trains and ships loading. In this case, straddle carriers start by picking-up a container from the yard and then drive to the delivery location (trucks areas, trains area or ship areas) to deliver it to their recipient. The third category of missions concerns the move of a container from a ship to another one. Finally, the last kind of missions concerns internal yard optimization process. Indeed, in some cases, it can be useful to reorganize a part of the stock area in order to reduce further delivery times or to free strategic container slots for next unloading missions.

Two time windows are affected to every mission. One concerns the pickup phase, the other one is related to the delivery. These time windows are used to fix an appointment between straddle carriers and hypothetical customer vehicles (trucks, trains or ships) concerned by the missions. Straddle carriers have to reach the pickup or delivery location within the given time window and so does the customers vehicles. If a straddle carrier comes too early, it will have to wait. On the contrary, if it comes too late, the customer vehicle will have to wait. As a consequence, a time window overrun implies a cost for the terminal because, if a customer has to wait excessively, it may require late fees from the container terminal exploitation company. However, in the case of yard optimization missions, the time windows can be overrun because it has no direct effect on the customers. So, according to the mission kind, time windows can be hard or soft. For incoming missions, the pickup time window is hard and the delivery time window is soft. For the outgoing container missions, the pickup time window is soft but the delivery time window is hard. For transhipment missions, both time windows are hard, and for yard optimization missions both time windows are soft. Those time windows characteristics have to be taken into account in the mission scheduling process (Balev, Guinand, Lesauvage and Olivier 2009).

2.5. Terminal de Normandie

The LDTT's laser system has been first implanted within the *Terminal de Normandie* (Normandy Terminal) in Le Havre. This is the reason why we have first modelled this terminal.

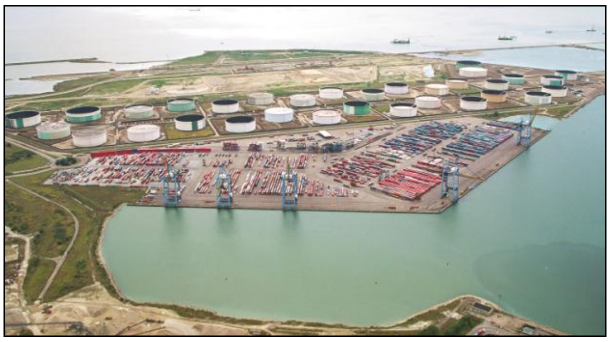


Figure 6: *Terminal de Normandie*, Harbour of Le Havre, France (source: http://www.t-n.fr).

It is delimited at north by the quay of Asia (North West) and the quay of Osaka (North East), and at South by the truck and train areas. This terminal is made up of 1170 crossroads, 170 roads, 531 lanes and 3499 containers slots. The yard contains 12 stocking areas and there are 3 train lanes and 3 truck areas. This terminal has been created in 1990 in order to contain the flow of container exchanges which has grown at a sustained rate. It is able to deal with the biggest container ships.

It uses five quay cranes and a fleet of about twenty straddle carriers. There is no stocking crane in this terminal. So the traffic regulation within the terminal is essential to ensure a sufficient quality of service for the customers. Moreover, if a straddle carrier comes too early at a truck or train location, it will have to wait for the truck or the train to pickup or deliver the container. The vehicles depot is located at the centre of the terminal to reduce the travelling costs to the containers locations. In this container terminal, the straddle carriers do not have to pass by the depot after every mission, but in order to avoid traffic jam, if a straddle carrier has no mission to accomplish, then it will go back to the depot.

In order to model this terminal, a detailed computerized plan has been created based on another rough plan supplied by our partners. The xml description file of the *Terminal de Normandie* resulting from this work contains around 2500 lines just to describe its structure and the road network.

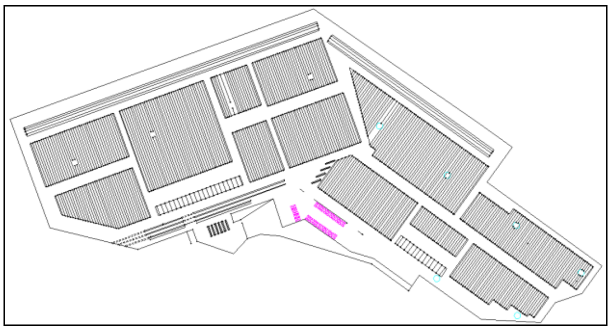


Figure 7: Detailed computerized plan of the *Terminal de Normandie*.

2.6. Dynamics modelling

Time modelling is essential in D²CTS because we aim at studying the dynamics impact on the system behaviour. We chose to use discrete time in order to be able to model events related to the dynamics. Therefore, discrete time allows setting up the time step depending on the desired level of precision and of performance.

As a matter of fact, a container terminal is a complex system made up of a large number of entities interacting with each other's and changing the state of the terminal at every moment. This system is also open, and external flows come in and out, making it to change. These flows concern trucks, trains and ship arrival/departure. Moreover dynamic events can occur within the container terminal such as handling vehicles or cranes failure, mission arrival into the system, mission cancelling, container lost, and every events generated by the human behaviour such as the failure to adhere to the mission schedule or the routing paths. The events are modelled by a start time stamp, a system knowledge time stamp and a description. When the simulation time reaches the time of a start time stamp, the event is triggered.

3. IMPLEMENTATION OF THE MODEL

3.1. What does D²CTS do?

D²CTS is able to model a container terminal and its dynamics. So it is able to perform some tests about:

- the terminal architecture;
- the terminal communication architecture;
- the vehicles routing;
- the vehicles fleet size;
- the mission scheduling;
- the containers location;
- the containers traffic absorption capacity.

Indeed, the program allows changing the configuration of the terminal itself. We can for instance decide to add a depot or to remove a stocking area and measure the consequences on the terminal evolution. It is also possible to add roads in the network or to remove some of them to test the impact of such modifications on the traffic within the terminal.

Moreover, several routing algorithms can be used. The Floyd-Warshall's and Dijkstra's algorithms (Floyd 1962, Dijkstra 1971) have already been implemented. Furthermore, we developed a routing algorithm based on the Dijkstra's algorithm, taking waiting time and FIFO arcs into account.

D²CTS can also be useful to perform some tests on the size of the straddle carriers' fleet. Indeed, a straddle carrier is very expensive and such an outlay has to be considered carefully. So, D²CTS can run two simulations: a first one without adding the new straddle carrier, and a second one with the new vehicle. Then, the both results can be compared to help the decision makers to reduce the risk factor related to such an investment.

In the same way, several mission scheduling policies or containers location strategies can be tested thanks to the simulator.

3.2. Technology and generic programming

D²CTS is written in Java which ensures a large flexibility in coding and executing the program. The distribution process uses the Java RMI technology and all the data are described by XML files. It concerns the terminal structure and road network, the vehicles characteristics, the containers location, the dynamic events and the program distribution.

The graphical user interface uses the GraphStream API (Dutot, Guinand, Olivier and Pigné 2007) which is an open-source library developed within the University of Le Havre and able to model and draw dynamic graphs. On the other hand, EADS Astrium, our partner within the CALAS project has developed a 3D graphical user interface able to simulate the driving of the straddle carriers. It uses data sent by D²CTS through a database to reproduce the container terminal structure and components. The actions realized within their program are also reproduced within D²CTS thanks to the database interconnection.

The program architecture is based on a modular paradigm. It means that the simulator contains a kernel connected or not to other parts. This approach has the benefits to ensure a high flexibility for developing new modules and for adapting the simulator to the users' expectations. Thus, it is possible to run D²CTS without graphical user interface or, for instance, to perform some tests with different routing algorithms.

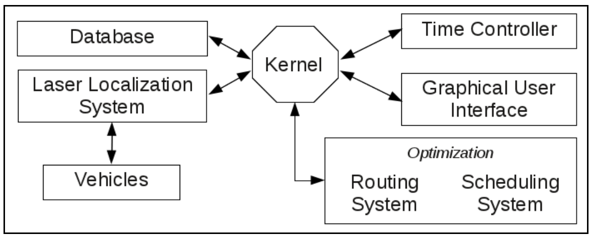
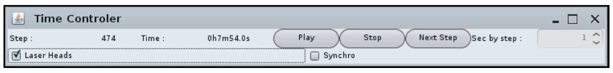


Figure 8: D²CTS architecture.

The available modules are:

- the kernel;
- the time controller;
- the 2D graphical user interface;
- the laser localization system;
- the vehicles routing algorithms;
- the missions scheduling algorithms;
- the automated vehicles component.


Time Controler

Step : 474 Time : 0h7m54.0s Sec by step : 1

[✓] Laser Heads [ ] Synchro

Figure 9: Time controller GUI snapshot.

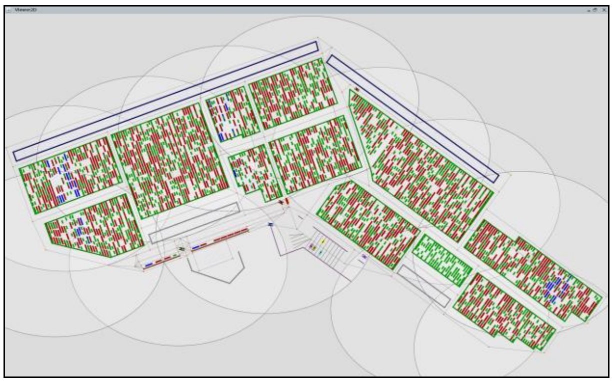
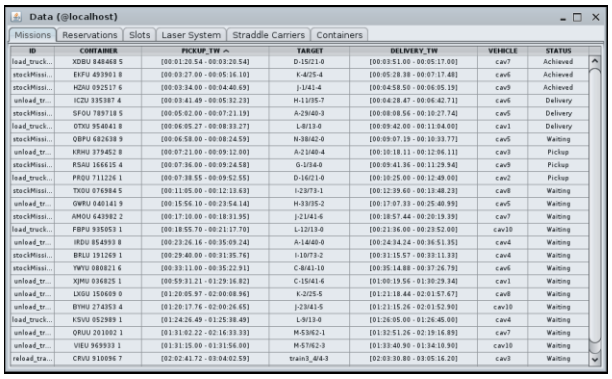


Figure 10: D²CTS 2D GUI snapshot of a simulation using the *Terminal de Normandie*.


Data (@localhost)

ID	CONTAINER	PICKUP_TW ^	TARGET	DELIVERY_TW	VEHICLE	STATUS
load_truck...	XDBU 848468 5	[00:01:20.54 - 00:03:20.54]	D-15/21-0	[00:03:51.00 - 00:05:17.00]	cav7	Achieved
stockMissi...	EKFU 493901 8	[00:03:27.00 - 00:05:16.10]	K-4/25-4	[00:05:28.38 - 00:07:17.48]	cav6	Achieved
stockMissi...	HZAU 092517 6	[00:03:34.00 - 00:04:40.69]	J-1/41-4	[00:04:58.50 - 00:06:05.19]	cav9	Achieved
unload_tr...	ICZU 335387 4	[00:03:41.49 - 00:05:32.23]	H-11/35-7	[00:04:28.47 - 00:06:42.71]	cav6	Delivery
stockMissi...	SFOU 789718 5	[00:05:02.00 - 00:07:21.19]	A-29/40-3	[00:08:08.56 - 00:10:27.74]	cav5	Delivery
load_truck...	OTXU 954041 8	[00:06:05.27 - 00:08:33.27]	L-8/13-0	[00:09:42.00 - 00:11:04.00]	cav1	Delivery
stockMissi...	QBPU 682638 9	[00:06:58.00 - 00:08:24.59]	N-38/42-0	[00:09:07.19 - 00:10:33.77]	cav5	Waiting
unload_tr...	KRHU 379452 8	[00:07:21.00 - 00:09:12.00]	A-21/40-4	[00:10:18.11 - 00:12:06.11]	cav3	Pickup
stockMissi...	RSAU 166615 4	[00:07:36.00 - 00:09:24.58]	G-1/34-0	[00:09:41.36 - 00:11:29.94]	cav9	Pickup
load_truck...	PRQU 711226 1	[00:07:38.55 - 00:09:52.55]	D-16/21-0	[00:10:25.00 - 00:12:49.00]	cav2	Pickup
stockMissi...	TXOU 076984 5	[00:11:05.00 - 00:12:13.63]	I-23/73-1	[00:12:39.60 - 00:13:48.23]	cav8	Waiting
unload_tr...	GWRU 040141 9	[00:15:56.10 - 00:23:54.14]	H-33/35-2	[00:17:07.33 - 00:25:40.99]	cav5	Waiting
stockMissi...	AMOU 643982 2	[00:17:10.00 - 00:18:31.95]	J-21/41-6	[00:18:57.44 - 00:20:19.39]	cav7	Waiting
load_truck...	FBPU 935053 1	[00:18:55.70 - 00:21:17.70]	L-12/13-0	[00:21:36.00 - 00:23:52.00]	cav10	Waiting
unload_tr...	IRDU 854993 8	[00:23:26.16 - 00:35:09.24]	A-14/40-0	[00:24:34.24 - 00:36:51.35]	cav4	Waiting
stockMissi...	BRLU 191269 1	[00:29:40.00 - 00:31:35.76]	I-10/73-2	[00:31:15.57 - 00:33:11.33]	cav4	Waiting
stockMissi...	YWYU 080821 6	[00:33:11.00 - 00:35:22.91]	C-8/41-10	[00:35:14.88 - 00:37:26.79]	cav6	Waiting
unload_tr...	XJMU 036825 1	[00:59:31.21 - 01:29:16.82]	C-15/41-6	[01:00:19.56 - 01:30:29.34]	cav1	Waiting
unload_tr...	LXGU 150609 0	[01:20:05.97 - 02:00:08.96]	K-2/25-5	[01:21:18.44 - 02:01:57.67]	cav8	Waiting
unload_tr...	BYHU 274353 4	[01:20:17.76 - 02:00:26.65]	J-23/41-5	[01:21:15.26 - 02:01:52.90]	cav10	Waiting
load_truck...	KSVU 052989 1	[01:24:26.49 - 01:25:38.49]	L-9/13-0	[01:26:05.00 - 01:26:45.00]	cav4	Waiting
unload_tr...	QRUU 201002 1	[01:31:02.22 - 02:16:33.33]	M-53/62-1	[01:32:51.26 - 02:19:16.89]	cav7	Waiting
unload_tr...	VIEU 969933 1	[01:31:15.00 - 01:31:56.00]	M-57/62-3	[01:33:40.90 - 01:34:10.90]	cav10	Waiting
reload_tra...	CRVU 910096 7	[02:02:41.72 - 03:04:02.59]	train3_4/4-3	[02:03:30.80 - 03:05:16.20]	cav3	Waiting

Figure 11: Snapshot of the missions tab of the D²CTS data view.

These modules can be executed on several and heterogeneous computers. The distribution process uses Java RMI technology which implies a serialization of the remote objects. But the data sent through the network can be weighty; this is the reason why the distribution configuration must be handled carefully. A bad configuration implies high communication times and the performance might drop.

3.3. Configuration

D²CTS uses XML files to describe its configuration. The files concern the network and the distribution configuration, the terminal description (stock and exchange areas, road network, vehicles depot, containers location...), the laser localization system description (laser heads location and range), and the vehicles description, location, and behaviour. Those files are then parsed by the simulator which creates corresponding objects according to the description.

3.4. Vehicles handling

In the simulator, a straddle carrier can be either autonomous or man-driven.

In the first case, the straddle carrier behaviour is handled by the software which checks objectives at each step of time. If the objective is not reached, then a direction is computed and the straddle carrier makes a move towards it. Else, a new objective is computed.

In the second case, the straddle carrier is driven by a user through the EADS Astrium 3D interface. So, the simulator supplies a mission workload, the routing paths to these locations, and just checks the new location of the straddle carrier trying to detect if the vehicle is following its given path. If an error is detected the path has to be recomputed. Moreover, if an event occurs concerning the road network or mission scheduling, new routing solutions are computed and sent to the concerned vehicles.

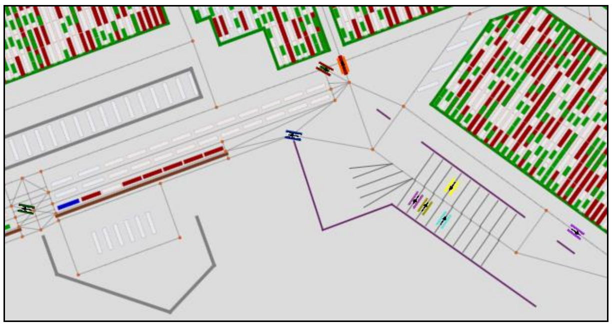


Figure 12: Straddle carriers 2D implementation in the simulator.

A straddle carrier can also generate some events such as failures or the impossibility to pick-up or to deliver a container at a given location. These events occur when the state of the terminal in the data structure does not fit with the reality. It may happen if a straddle carrier driver chose a mission and decide to change but without advising the system. So, the system becomes corrupted and the errors can be detected much later. D²CTS aim at detecting incoherencies between a supposed behaviour of a straddle carrier and its real actions. If such incoherencies are detected, the system tries to repair them if it is possible. Otherwise, a communication between the straddle carrier driver and the control operator is required to take decisions and acting on either the straddle carrier activity or the system knowledge of the terminal state.

3.5. Test data generation

Another module has been developed within D²CTS in order to generate realistic data. It concerns the starting state of a terminal and events generation.

The container generator is used to create containers and to locate them within a given terminal. The algorithm makes sure that the container can physically be placed at the computed location. The corresponding XML file is then generated.

The mission generator role is to provide missions events within a given period of time and according to an initial state of a terminal. It generates ships, trucks and trains arrival and departure. The algorithm makes sure that the vehicles location will be free at the arrival time. Next, it computes which containers have to be loaded or unloaded. Finally, the time windows for both pickup and delivery operations are calculated by taking into account the travel time between those locations. A time margin is next added to let a relative tolerance to fit with reality constraints. The events are finally written in a XML file which can be added for further simulations.

4. CONCLUSION AND PERSPECTIVES

D²CTS is a container terminal simulator able to reproduce dynamic events and to perform tests concerning optimization matters such as the terminal architecture, the vehicles routing or the mission scheduling. It is fully adaptable thanks to a XML data description structure and distribution configuration.

The calibration of D²CTS will consist in representing the state of the terminal at the very beginning of a working day. Then, the validation will consist in running the simulation taking into account the real events occurring in the working day at the container terminal and finally by comparing the state of the real terminal and of the simulated one at the end of the day.

We are currently developing routing algorithms taking into account the graph properties of a container terminal road network. We are also creating mission scheduling policies based on meta-heuristics. Those problems are defined as NP-hard (Bish, Leong, Li, Ng and Simchi-Levi 2001) and only a simulation approach could propose an integrated solution for all the concerned subsystems (Soriguera, Espinet and Robuste 2006). The next step of our work will consist in measuring the performance of these algorithms thanks to D²CTS.

ACKNOWLEDGMENTS

We would like to thank our sponsors in the CALAS project: the D.G.E (*Direction Générale des Entreprises*) and the *Région Haute Normandie* for their financial support. We also wanted to thank our partners: LDTT, *Ateliers de Normandie*, EADS Astrium, LMAH and INSA of Rouen.

REFERENCES

Balev, S., Guinand, F., Lesauvage, G. and Olivier, D., 2009, Dynamical handling of straddle carriers activities on a container terminal in uncertain environment - a swarm intelligence approach. In *ICCSA 2009, The 3rd International Conference on Complex Systems and Applications*, June 29th – July 2nd, Le Havre, France.

Bish, E. K., T. Y. Leong, C. L. Li, J. W. C. Ng, D. Simchi-Levi, 2001, Analysis of a new vehicles scheduling and location problem. In Naval Research Logistics, Vol. 48, 2001, pp. 363-385.

Cavin, D., Sasson, Y. and Schiper, A. 2002, On the accuracy of MANET simulators. In proceedings of *the 2nd Int'l Workshop on Principles of Mobile Computing*, ACM Press, Toulouse, France, pp.38–43.

Dijkstra, E.W., 1971, EWD316: A Short Introduction to the Art of Programming. T. H. Eindhoven, The Netherlands, August.

Dutot, A., Guinand, F., Olivier, D. And Pigné, Y., 2007, Graphstream: A tool for bridging the gap between complex systems and dynamic graphs. In *Emergent Properties in Natural and Artificial Complex Systems. Satellite Conference within the 4th European Conference on Complex Systems (ECCS'2007)*, October 4th – 5th, Dresden, Germany.

Floyd, R.W., 1962, Algorithm 97: Shortest path. In *Communications of ACM,* volume 5, number 6, p345.

Fujimoto, R. M., 2000, Parallel and Distributed Simulation Systems, *Wiley Interscience*.

Heidemann, J., Bulusu, N., Elson, J., Intanagonwiwat, C., Lan, K., Xu, Y., Ye, W., Estrin, D. and Govindan, R., 2001, Effects of detail in wireless network simulation. In proceedings of *SCS Multiconference on Distributed Simulation*.

Lesauvage, G., Balev, S. and Guinand, F., 2011, Routage des chariots cavaliers sur un terminal portuaire à conteneurs. In ROADEF 2011, *Société française de Recherche Opérationnelle et Aide à la Décision*, March 2nd-4th, Saint-Etienne, France.

Orda, A. and Rom, R., 1990, Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. In *Journal of the ACM*, volume 37 issue 3: 607-625.

Soriguera, F., Espinet, D., Robuste, F., 2006, A simulation model for straddle carrier operational assessment in a marine container terminal. In *J Maritime Res* 3(2):19–34.

Steenken, D., Voss, S. And Stahlbock R., 2004, Container terminal operation and operations research – a classification and literature review. In *OR Spectrum* 26: 3-49.