Exploiting the easyABMS methodology in the logistics domain

Alfredo Garro, Wilma Russo

***Abstract*— ABMS (Agent-Based Modeling and Simulation) has arisen as new approach to effectively support domain experts to cope with the growing complexity of the problems which they have to face and solve. To date, few methodologies are available which can be exploited by domain experts with limited programming expertise to model and subsequently analyze complex systems typical of their application domains. The easyABMS methodology has been proposed to overcome the lack of integrated methodologies able to seamlessly guide domain experts from the analysis of the system under consideration to its modeling and analysis of simulation results. In this paper, the effectiveness of easyABMS is demonstrated through a case study in the logistics domain which concerns the analysis of different policies for managing vehicles used for stacking and moving containers in a transshipment terminal.**

Index Terms— **Agent-Based Modeling and Simulation, Agent-Oriented Methodologies, Container Terminal Management.**

I. Introduction

Agent Based Modeling and Simulation (ABMS) is a new approach for analyzing and modeling complex systems, an approach which is becoming acknowledged for its efficacy in several application domains (financial, economic, social, logistics, physical, chemical, engineering, etc) [17]. ABMS, allows for the definition of a system model based on autonomous, goal-driven and interacting entities (agents) organized into societies which is then simulated so to obtain significant information on not only the properties of the system under consideration but also its evolution.

Although several ABMS tools are currently available [10, 11, 19, 20, 24], there are only a few methodologies involving well- defined processes which are able to cover all the phases from the analysis of the system under consideration to its modeling and subsequent analysis of simulation results [7, 8, 17]. As a result, simulation models are often obtained using the two following approaches: (i) a direct implementation based on a chosen ABMS tool of the simulation model whose abstraction level is then too low and platform dependent as a conceptual modeling phase is not available; (ii) adapting a given conceptual system model to a specific ABMS tool which, however, requires additional adaptation, calling for extra work, the amount of which increases depending on the gap between the conceptual and the implementation model of the system. Thus, both approaches lead to simulation models which are difficult to verify, modify and update.

To address these issues, a new methodology, easyABMS, has recently been proposed [4,5] which has specifically been conceived for agent-based modeling and simulation of complex system, seamlessly covering all the phases from the analysis of the system under consideration to its modeling and analysis of simulation results. easyABMS defines an *iterative* process which is *integrated*, *model-driven* and *visual*. In particular, each phase of the process refines the model of the system which has been produced in the preceding phase and its work-products are mainly constituted by *visual* diagrams based on the UML notation [23]. In addition, according to the *model-driven* paradigm [1, 21] the simulation code is automatically generated from the derived system Simulation Model. On the basis of the simulation results, a new/modified and/or refined model of the system can be obtained through a new process iteration which can involve all or some process phases.

Currently, easyABMS exploits the advanced features of visual modeling and of (semi)automatic code generation provided by the *Repast Simphony Toolkit* [17,18], a very popular and open source ABMS platform.

In this paper, the effectiveness of easyABMS in supporting domain experts to fully exploit the benefits of the ABMS, while significantly reducing programming and implementation efforts, is exemplified through a case study in the logistics domain. Specifically, the case study is focused on the analysis of different policies for the management of vehicles used for stacking and moving containers (*straddle carriers*) in a container transshipment terminal.

The remainder of this paper is organized as follows: Section II presents an overview of the easyABMS methodology and the related process; Section III presents a brief introduction to the reference application domain (a container transhipment terminal) and to related management problems; Section IV shows the application of easyABMS to the agent-based modeling and simulation of the Straddle Carrier Routing and Dispatching problem; finally, conclusions are drawn and future works delineated.

A. Garro is with the Department of Electronics, Informatics and Systems (DEIS), University of Calabria, Rende (CS), 87036 Italy. (e-mail: alfredo.garro@unical.it).

W. Russo is with the Department of Electronics, Informatics and Systems (DEIS), University of Calabria, Rende (CS), 87036 Italy. (e-mail: w.russo @unical.it).

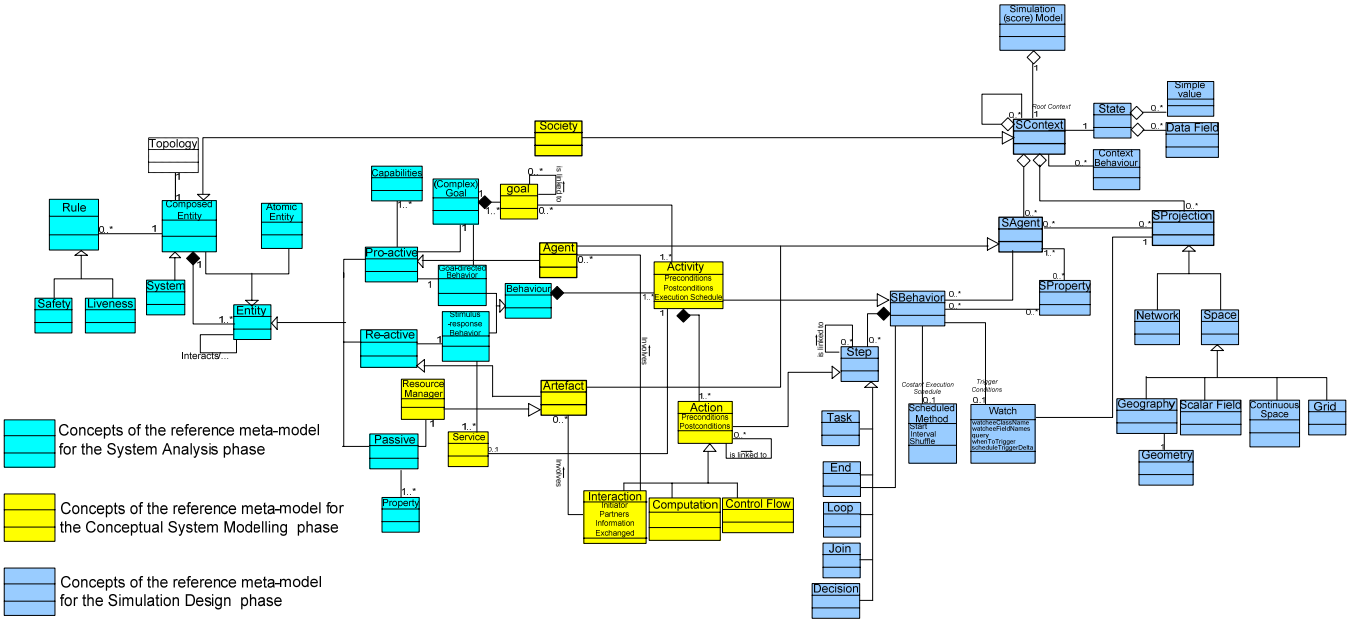


Fig. 1. The reference meta-model of easyABMS.

II. An overview of easyABMS

The easyABMS methodology defines an iterative process for ABMS composed of seven subsequent phases from the *System Analysis* to the *Simulation Result Analysis* [4, 5]. On the basis of the simulation results obtained a new iteration of the process which can involve all or some process phases can be executed for achieving new simulation objectives or those which have not yet been obtained. Specifically, the process phases are the following:

- *System Analysis*, in which a preliminary understanding of the system and the main simulation objectives are obtained (Analysis Statement);
- *Conceptual System Modeling*, in which a model of the system is defined in terms of agents, artifacts and societies (Conceptual System Model);
- *Simulation Design*, in which a model of the system is defined in terms of the abstractions offered by the framework which is exploited for the simulation (Simulation Model);
- *Simulation Code Generation*, in which the Simulation Code for the target simulation environment is automatically generated starting from the model which is obtained in the previous phase;
- *Simulation Set-up*, in which the Simulation Scenarios are established;
- *Simulation Execution* and *Results Analysis*, in which the simulation results are analyzed with reference to the objectives of the simulation previously identified in the *System Analysis* phase.

Currently, all the simulation related phases are supported by the *Repast Simphony Toolkit* [18, 20]. In particular, the *Simulation Design* and the *Simulation Code Generation* phases are supported by the *Repast Simphony Development Environment* [15], while the *Simulation Set-up*, the *Simulation Execution* and the *Simulation Results Analysis* phases are supported by the *Repast Simphony Runtime Environment* [16].

The models of the system generated by each process phase are produced according to the well-defined reference meta-model shown in Figure 1 so to facilitate the verification of the correctness of the models produced. Moreover, the concepts related to each phase are defined by extending and/or refining those of the previous phase; this allows for the seamless integration between the phases as the model produced in each phase extends and/or refines the model of the system produced in the previous phase.

The following sub-sections provides a brief description of each process phase.

A. System Analysis

In the *System Analysis* phase, the objectives of the simulation are specified and a preliminary understanding of the system and its organization is obtained.

This phase is based on the principle of *layering*, exploiting the well-known techniques of *Decomposition*, *Abstraction* and *Organization* [2, 9], and is constituted of a sequence of *analysis steps*. In each *step* a new system representation is produced by applying the *in-out zooming mechanisms* [12] to the entities comprising the system representation which resulted from the preceding *analysis step*. In the first *analysis step*, a starting level of abstraction for analyzing the system is chosen and then the system is *zoomed-in* so to identify its

component entities on the basis of the starting abstraction level.

According to the reference meta-model of the *System Analysis* phase (see Fig. 1), an *Entity* can be characterized by autonomous and goal-oriented behavior (*pro-active entity*), purely stimulus-response behavior (*re-active entity*), or can be *passive*. In addition, both the rules governing entities and their evolution, and the relationships among entities are specified. Specifically, *Safety* rules determine the *acceptable and representative* states of an entity whereas *liveness* rules determine which state transitions are feasible during entity evolution. Relationships can be either *intra-entity relationships* (i.e. relationships among the component entities obtained by the zooming-in of an entity) or *inter-entity relationships*.

The *System Analysis* phase ends when the user obtains a *System Representation* in which each component (pro-active, re-active, passive) entity has been represented at the level of abstraction which is appropriate for the objectives of the simulation. This *System Representation*, along with a synthetic description of the system being considered, a detailed description of each identified entity and the objectives of the simulation, constitutes the work-product of this phase (the ***Analysis Statement***).

B. Conceptual System Modeling

In the *Conceptual System Modeling* phase, the *Structural System Model* is produced, and in particular, for each entity in the *System Representation*:

- the abstraction level suited to specific simulation objectives is chosen;
- the conceptual representation , in terms of *Agent*, *Artifact* or *Society*, is derived on the basis of the associations among the main concepts of the *System Analysis* and *Conceptual System Modeling* phases (see Fig. 1);
- the *interactions* with the other entities are obtained from the *intra* and *inter-relationships* where the latter cross the boundaries of societies.

The chosen level of abstraction of an entity can be modified in successive iterations through which it is then possible to produce new, modified, and/or refined *Structural System Model*s.

For each entity in the produced *Structural System Model* a specific model is then defined, whose type can be one of the following depending on the entity type:

- ***Society Model*** which describes the entities which compose a *Society*, their type (*Agent*, *Artifact*, *Society*), and the rules governing the *Society* (*safety* rules) and its evolution (*liveness* rules);
- ***Agent Model*** which details the complex goal of an *Agent* (***Agent Goal Model***), its behavior as a set of *periodically scheduled* and *triggered Activities* (i.e. flow of *Actions*) which contribute to the achievement of the *Agent* goals (***Agent Behavioral Model***), and its interactions with other *Agents* and *Artifacts* in which the agent is involved (***Agent Interaction Model***);
- ***Artifact Model*** which describes the behavior of an *Artifact* as a set of *triggered Activities* related to the offered services (***Artifact Behavioral Model***), and its interactions with other *Artifacts* and *Agents* (***Artifact Interaction Model***).

C. Simulation Design

In this phase, starting from the *Conceptual System Model* a *Simulation Model* of the system, in terms of the abstractions offered by the framework exploited for the simulation, is produced.

In Figure 1 the basic simulation concepts of the reference simulation framework (the *Repast Simphony Toolkit* [18, 20]) are highlighted. Specifically, the central concept is the (simulation) *Context* (*SContext*) which represents an abstract environment in which (simulation) *Agents* (*SAgent*s) can act and is provided with an internal *state* consisting of *simple values* and *Data Fields* (a n-dimensional field of values). In addition, an *SContext* can also support *behaviors* for the management of its internal *state*. *SContexts* can be organized hierarchically so to contain *sub-SContexts* which can have their own state. *SAgents* in an *SContext* can be organized by using *Projections* which are structure designed to define and enforce relationships among the *SAgents* in the *SContext*. In particular, a *Network Projection* defines the relationships of both acquaintance and influence between *SAgents* whereas *Space Projections* define (physical or logical) space structures (*Grid, Scalar Fields, Continuous Space, Geography*) in which the agents can be situated.

An *SAgent* can have multiple *behaviors* (*SBehavior*s), each operating on *SAgent Properties* and consists of a sequence of *Steps*; each *Step* can be associated with the execution of a *Task* or with the control of the flow of the *Task* execution (*Loop, Join, Decision, End*). Each *SBehavior* can be characterized by a *Scheduled Method* which defines a constant execution schedule, and by a *Watch* which periodically, on the basis of some *watched* parameters and conditions, triggers the execution of the behavior.

A *Repast Simphony* simulation model is defined by first specifying the structure and the characteristics of the *root SContext* and of all the possible nested *sub-SContexts*, in terms of their components (*SAgents*, *Projections* and *sub-SContexts*), and, then, specifying for each *SAgent* its *Properties* and *SBehaviors*, and for each *SBehavior* the component *Steps*, and the associated *Scheduled Method* and *Watch*.

The associations among the above described simulation concepts of the *Repast Simphony Toolkit* and the related concepts of the *Conceptual System Model* are reported in Figure 1. The exploitation of these associations makes it possible to directly obtain, starting from the *Conceptual System Model*, the *Simulation Model* of the System as follows:

- each *Society* becomes a Repast Simulation Context (*SContext*), the System is the root *SContext* and any enclosed *Society* is a (sub)-*Context* of the corresponding enclosing *Society*;

– *Artifacts* and *Agents* become Repast Simulation Agents (*SAgents*), the *Activities* which constitutes their behaviors are easily converted into Repast Simulation Behaviors (*SBehaviors*);
– *relationships* derived from *Interactions* among *Agents* and *Artifacts* generate *Repast Network Projections.*

D. The other Simulation related phases

According to the Model Driven paradigm [1, 21], the *Repast Simphony Development Environment* [15] is able to automatically generate a great part of the simulation code from the derived *Simulation Model* of the system. The simulation which can be extended with additional Java and XML code is then compiled by the *Repast Simphony Development Environment* using a Java compiler and then loaded into the *Repast Simphony Runtime Environment.*

The simulation executed by the *Repast Symphony Runtime Environment* can start after establishing: (i) the simulation scenario by specifying the values of the simulation parameters defined in the *Simulation Design* phase; (ii) the presentation preferences for the simulation results concerning the system properties of interest identified during the *Simulation Design* phase.

Finally, the obtained simulation results can also be analyzed by exploiting the analysis tools (Matlab, R, VisAd, iReport, Jung) which can be directly invoked from the *Repast Simphony Runtime Environment* so to verify whether the objectives of the simulation identified during the *System Analysis* phase have been achieved. Where objectives have not been achieved or where new simulation objectives emerge, a new iteration of the process can be executed, which can then involve all or some process phases so that new/modified and/or refined models of the system can be produced for achieving the remaining/new simulation objectives.

TABLE I
MANAGEMENT PROBLEMS IN CONTAINER TRANSHIPMENT TERMINALS

PHASE	PROBLEM	DESCRIPTION
Arrival of the containership	QUAY CRANE ASSIGNMENT PROBLEM (QCAP)	Determining the number of quay cranes to assign to an incoming vessel.
	BERTH ALLOCATION PROBLEM (BAP);	Assigning incoming ships to berths, by taking into account constraints in both spatial and temporal dimensions so to minimize the time each ship spends in port (*turnaround time*).
Unloading and Loading of the ship	QUAY CRANE SCHEDULING PROBLEM (QCSP)	Determining a sequence of unloading and loading movements for cranes assigned to a vessel in order to minimize the vessel completion time as well as the crane idle time.
Transport of containers from the ship to the yard and vice versa	YARD MANAGEMENT	Allocating and reallocating the containers in the yard in order to reduce the amount of time required to handle of each vessel.
	STRADDLE CARRIER ROUTING AND DISPATCHING (SCRD)	Determining the operation to be performed by the straddle carries to maximize the productivity of each crane.

III. MANAGEMENT OF A CONTAINER TRANSHIPMENT TERMINAL

Due to the continuous growth in the volume of goods exchanged around world, further boosted by the rising Chinese and Indian economies, maritime transportation is becoming a crucial asset in global economy as it allows for large economies of scale in the transport sector. Specifically, the current maritime transportation system is based on a hub and spoke model [22] whereby ultra-large containerships operate between a limited number of mayor (mega)transhipment terminals (hubs), and smaller vessels (feeders) which link the hubs with other minor ports (spokes).

In this scenario, a hub terminal must maintain a high level of efficiency, not only to avoid traffic congestion but also to increase its competiveness as some main characteristics (geographical, structural and technological) which also determine the competitiveness of a container terminal can be modified only on a long term perspective.

It thus becomes crucial to increase hub efficiency, rendering it more competitive through the optimal management of terminal resources and optimizing tactical and operational logistics.

In the next sub-section, the organization of a maritime container terminal and some primary management issues are briefly discussed; a more complete description can be found in [13].

A. Organization of a Container Transhipment Terminal

Each ship approaching a maritime terminal enters in a harbour and waits to moor at an assigned berth position along the terminal quay which is equipped with giant cranes (quay cranes) for loading and unloading containers. These containers, in a *DTS* (*Direct Transfer System*) terminal, are transferred to and from the terminal yard by a fleet of vehicles (*straddle carrier*) which are able to stack containers in the yard. In contrast, in an *ITS* (*Indirect Transfer System*) terminal, containers are moved by trucks and trailers from the quay to the yard and vice-versa and staked by yard cranes.

In this context, the main logistic processes and related management problems can be grouped in relation to the flow

of containers in the terminal as shown and briefly described in Table 1; other issues are related to inter-terminal transportation and to possibly link with other transportation modes. Moreover, a transversal issue is related to the human resources management [13].

These very fundamental issues are not only reciprocally related, but the large-scale nature of hub management makes the use of standard exact solution algorithms impractical. In fact, the management of such large and intricately complex systems require new modeling methods which must also generate *proof-of-concept* simulations.

In the following Section, the effectiveness of the ABMS approach and the easyABMS methodology is shown focusing on the Straddle Carrier Routing and Dispatching Problem (SCRDP) [14]; with reference to the different management problems in a Container Transhipment Terminal (see Table 1), a more complete and domain specific agent-based simulator has been proposed in [6].

IV. Modeling and Simulating Straddle Carrier Routing and Dispatching through easyABMS

A. System Analysis

The main indicator of optimal performance in a container transhipment terminal is the *average ship-turn-around time* which is the average time-lapse between a ship's arrival and its departure, starting from the amount of time the ship waits for a berth (*berth waiting time*) and the duration for which the ship is docked for unloading and loading operations (*handling time*). In the following, the focus is set on the *handling time* given to fact that this time is highly dependent on the productivity of the Quay Cranes (QCs) and, as a consequence, on the management policies of the Straddle Carriers (SCs).

Specifically, to maximize the productivity of the QCs in a DTS container terminal, the SCs should operate so that the buffer of each crane, which has a limited capacity of only a few containers, is not full /not empty if the crane is performing the discharging/loading phase. Specifically, there are two main policies for organizing the work of SCs:

- *dedicated modality*: a given number of SCs are allocated to each QC to follow its working phases;
- *shared modality* (or *pooling*): a group of SCs is shared by two or more QCs which work on the same ship or on adjacent berthed ships and, possibly, frequently swapping between the tasks of loading and discharging containers.

The *shared modality* presents several benefits with respect to the *dedicated* mode: (i) reduction in the number of empty trips done by the SCs (i.e. travels without carrying any container), as the SCs can fruitfully alternate between trips carrying containers from the yard to the cranes which are loading outgoing cargo and trips back to the yard, carrying discharged cargo; (ii) more constant value of productivity of both QCs and SCs as, when a crane is not working, the SC of a pool can speed up operations of the other QCs.

A quantitative evaluation of the aforementioned benefits is not easy to obtain through traditional analytical models. Moreover, classical dispatching models [14] often fail to provide dynamic assignment of container moves to SCs of a pool in order to speed up the loading/discharging operations (the Straddle Carriers Pooling Problem - SCPP). To overcome these shortcomings, an agent-based model can be defined and simulated with the following main objectives:

(i) quantifying the benefits of the *pooling* modality with reference to system productivity (vessels handling time) and cost reduction (numbers of exploited SCs and total distance covered);

(ii) obtaining an effective solution for the dynamic assignment of container moves to the SCs of a pool which can be used for automatically drive the coordinated behavior of the SCs in a real container terminal.

The *System Representation* obtained on the basis of the identified simulation objectives is reported in Figure 2. All the entities represented in Figure 2 are further described, along with their relationships and their *safety* and *liveness* rules, in a textual format enriched by tables and diagrams which are not reported due to space limitations.

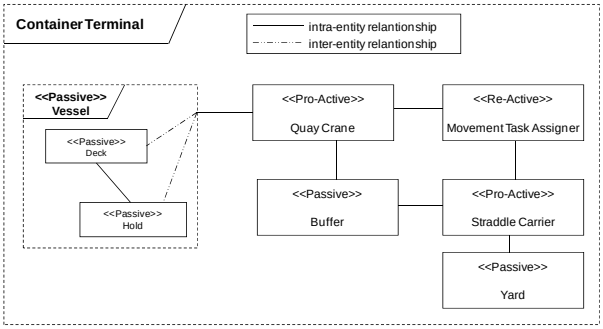


Fig. 2. System Representation

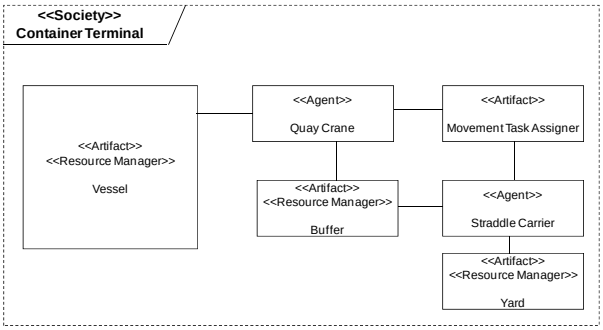


Fig. 3. Structural System Model

B. Conceptual System Modeling

The *Structural System Model* derived from the *System Representation* is reported in Figure 3; in particular, as the simulation objectives concern management policies of SCs, the level of representation chosen for the *Vessel* is more abstract with respect to the level resulting from the Analysis phase.

For each entity in the *Structural System Model* the corresponding *Society, Agent or Artifact Model* is defined (see Section II.B). Due to space limitations, the following sub-sections report only the *Society Model* for the *Container*

Terminal Society, the *Agent Model* for the *Straddle Carrier* Agent and the *Artifact Model* for the *Movement Task Assigner* Artifact.

1) The Container Terminal Society Model

The *Society Model* of the *Container Terminal Society* is shown in Figure 4 which reports the different entities which compose the *Society*, the *safety* and *liveness* rules which govern it and its dynamics.

Entity	Type
Vessel	Artifact (Resource Manager)
Quay Crane (QC)	Agent
Buffer	Artifact (Resource Manager)
Straddle Carrier (SC)	Agent
Movement Task Assigner	Artifact
Yard	Artifact (Resource Manager)

Safety rules

S_CTerm1. $NCvi(t) = NCvi(t_0) - NCDvi(t) + NCLvi(t)$;

where NCi(t) is the number of containers on the Vessel i at time t; NCDvi(t) is the number of containers that have been discharged from the Vessel i up to time t; NCLvi(t) is the number of containers that have been loaded onto the Vessel i up to time t.

S_Term2. ...

Liveness rules

L_CTerm1. A Quay Crane cannot download a container on its buffer if the buffer is full.

L_CTerm2. …

Fig. 4. The *Society Model* of the *Container Terminal Society*.

2) The Straddle Carrier Agent Model

Part of the *Agent Model* of the *Straddle Carrier* Agent is shown in Figure 5. In particular:

- Figure 5.a shows the *Straddle Carrier Goal Model* in which, as the two goals (*Movement of containers from Buffer to Yard* and *Movement of containers from Yard to Buffer*) can be achieved independently, no achievement relationship is present;
- Figures 5.b illustrates a part of the *Straddle Carrier Behavioral Model*; in particular, the *Straddle Carrier Activity Table* specifies the activities (*Container Movement Activity)* which the *Straddle Carrier* Agent executes for achieving its goals, along with the pre and post conditions and the execution schedule (periodical). Moreover, as the definition of an *Agent Behavioral Model* requires that each activity in the *Agent Activity Table* must be further described by an UML [23] *Activity Diagram*, the diagram for the *Container Movement Activity* is also shown. The UML *Activity Diagram* must be further enriched with an *Activity Action Table* (not shown in figure due to space limitations) which reports, for each single component action, a synthetic description of the action along with its pre and post conditions, the capabilities required for carrying out the action and its type (*computation or interaction*).
- Figure 5.c reports the *Straddle Carrier Interaction Model* which specifies, for each action of the *interaction* type (*Task Assignment Request*, *Assigner Response*) of the *Container Movement Activity*, the *initiator*, the *partners* of the interaction and the *exchanged information*.

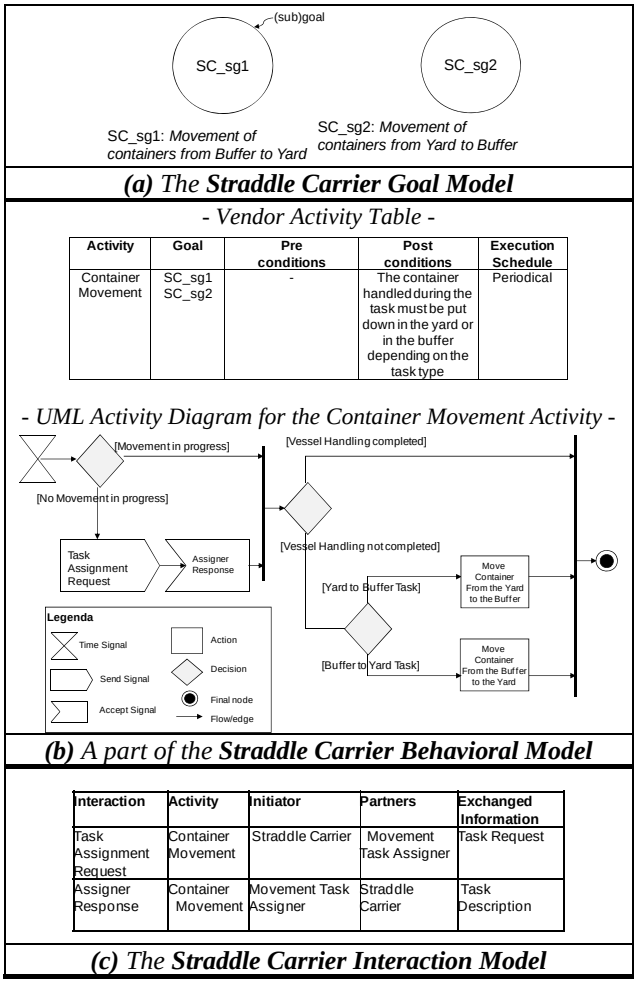


Fig. 5. Part of the *Agent Model* of the *Straddle Carrier Agent*.

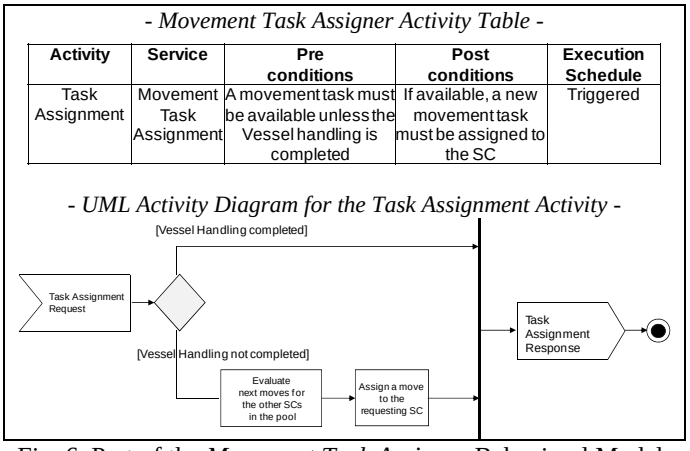
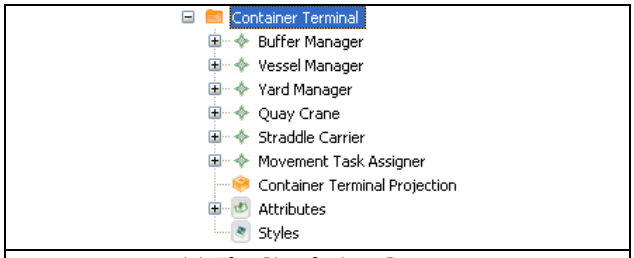


Fig. 6. Part of the *Movement Task Assigner* Behavioral Model.

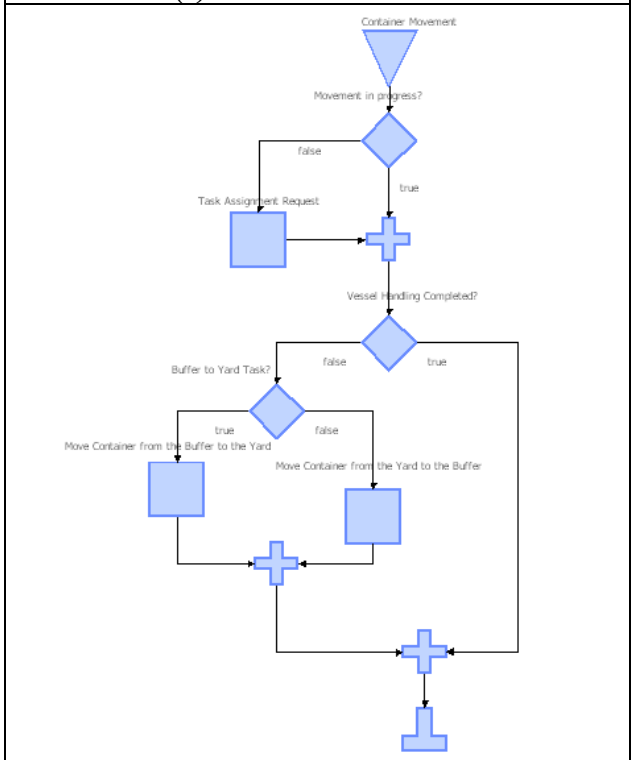
3) The Movement Task Assigner Artifact Model

Figure 6 presents part of the *Artifact Model* of the *Movement Task Assigner* Artifact, and, in particular, the part of the *Movement Task Assigner Behavioral Model* which describes the *Task Assignment Activity* triggered by an SC requesting a new container movement to be performed. In particular, at the completion of its container movement the SC requests the next assignment from the *Movement Task Assigner* (see Figure 5.c). The *Movement Task Assigner* must

then decide, from available moves, the next best move for the requesting SC taking into account also subsequent moves which could be assigned to the other SCs in the pool (*Lookahead Policy*). Such planning could be dynamically revised at the next task assignment request.



(a) The Simulation Context



(b) The Container Movement SBehavior of the Straddle Carrier SAgent

Fig. 7. Part of the *Simulation Model*.

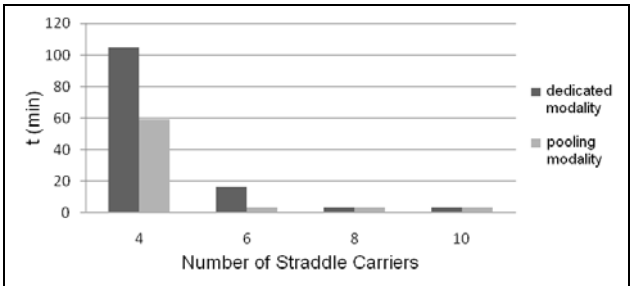
C. Simulation Design

Figures 7.a-b show a portion of the *Simulation Model* produced by adopting the *Repast Simphony Toolkit* [18, 20] as the reference simulation framework. Figure 7.a shows the organization of the Simulation Context *(SContext)* whereas Figure 7.b shows a Simulation Behavior *(SBehavior)* of the *SAgent* representing a *Straddle Carrier*. In particular, the *Container Movement SBehavior* in figure 7.b corresponds to the *Container Movement Activity* reported in figure 5.b. The seamless transition between the two models is highlighted by the comparison between these two figures which clearly demonstrates that the behavior of an Agent/Artifact, defined during the Conceptual Modeling phase in terms of *Activities* expressed by using the UML notation, can be directly mapped onto that of an *SAgent*, defined during the *Simulation Design* phase in terms of *SBehaviors.*

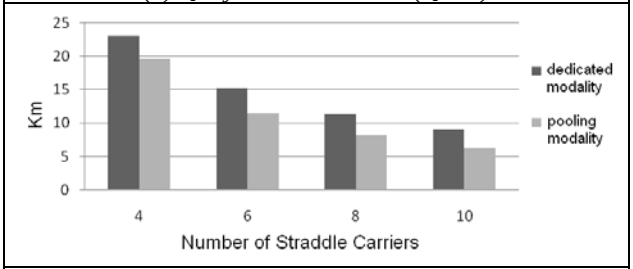
D. Simulation Execution and Results Analysis

Starting from the *Simulation Model* a great part of the simulation code is automatically generated by the *Repast Simphony Development Environment* [15], compiled by using a Java compiler and then loaded into the *Repast Simphony Runtime Environment* for the *Simulation Set-up* and *Execution*.

According to the simulation objectives, the execution of the resulting *Simulation Model* made it possible to compare and quantify the benefits of both *dedicated* and *pooling* modalities. In particular, several simulations have been executed for different scenarios in order to evaluate: the Quay Crane Idle Time (QCIT), the Straddle Carrier Covered Distance (SCCD), and the Straddle Carrier Idle Time (SCIT). As an example, Figures 8.a-b illustrate the QCIT and the SCCD, in the two different modalities, with reference to a simulation scenario based on real-life organizational topology and equipment typologies of the Gioa Tauro Container Terminal [3]. In this simulation scenario one Vessel is handled by two QCs for the loading and discharging of 50 containers respectively. The results shown in Figure 8, which are results averaged from 30 simulation runs, made it possible to quantify the significant advantage of the *pooling* modality in terms of vessel handling time and cost reduction.



(a) Quay Crane Idle Time(QCIT)



(b) Average Distance Covered by the Straddle Carriers

Fig. 8. Some Simulation Results.

V. CONCLUSION

Several tools for ABMS are now available as well as methodologies for the development of agent-based systems which are mainly proposed in the context of Agent-Oriented Software Engineering (AOSE). Nonetheless, only a few results are available which integrate the methodological

features coming from the AOSE with the modeling and simulation features of modern ABMS tools. As a consequence, scarce support in the whole process which goes from the system analysis to the analysis of simulation results is provided to domain experts with limited programming expertise. To address these issues, easyABMS, a recently proposed and full-fledged methodology for agent-based modeling and simulation of complex systems, fruitfully exploits both AOSE modeling techniques and simulation tools specifically conceived for ABMS.

In this paper, the effectiveness of easyABMS has been demonstrated using a case study in the logistics domain which concerns the analysis of different policies for managing *Straddle Carriers* in a *Container Transshipment Terminal*. In particular, overcoming the main limitations when using only classical analytical models, a quantitative assessment of two primary *Straddle Carrier* management policies and an effective solution in guiding the dynamic assignment of container moves have been easily provided. The exploitation of easyABMS allowed to demonstrate how this new methodology can seamlessly guide domain experts from the analysis of the system under consideration to its modeling and simulation, as the phases which compose the easyABMS process, the work-products of each phase, and the (seamless) transitions among the phases are fully specified. In addition, easyABMS focuses on system modeling and simulation analysis rather than details related to programming and implementation as it exploits the Model Driven paradigm, making it possible the automatic code generation from a set of (visual) models of the system.

Future research efforts will be devoted to: (i) extend the Repast Simphony Toolkit so to obtain an integrated ABMS environment which fully supports all the process phases also comprising the System Analysis and Conceptual System Modeling phases; (ii) extensively experiment easyABMS in case studies of social, financial, economic, and logistic relevance; (iii) adopting a meta-simulation framework for the Simulation Design phase so to obtain a platform-independent simulation model which can then be translated into different platform-dependent simulation models.

REFERENCES

[1] C. Atkinson and T. Kühne. Model-driven development: A metamodeling foundation. IEEE Software, 20(5):36-41, 2003.

[2] G. Booch. Object-Oriented Analysis and Design with Applications. Addison-Wesley, 1994.

[3] J.-F. Cordeau, M. Gaudioso, G. Laporte, and L. Moccia. The service allocation problem at the Gioia Tauro maritime terminal. European Journal of Operational Research, 176:1167–1184, 2007.

[4] A. Garro, W. Russo. An integrated agent-based process for the simulation of complex systems. Proceedings of the International Conference on Economic Science with Heterogeneous Interacting Agents (ESHIA), Warsaw, Poland, 19-21 June, 2008.

[5] A. Garro, W. Russo. An integrated and iterative process for agent-based modeling and simulation. Proceedings of the 5th International Conference of the European Social Simulation Association (ESSA), Brescia, Italy, 1-5 September, 2008.

[6] L. Henesey, P. Davidsson, and J. A. Persson. An agent-based approach for building complex software systems. Autonomous Agents and Multi-Agent Systems, 18(2):220-238, 2009.

[7] T. Iba and N. Aoyama. Understanding Social Complex Systems with PlatBox Simulator. In Proc. of the 5th International Conference on Computational Intelligence in Economics and Finance (CIEF2006), pages 64-67, Taiwan, October 2006.

[8] T. Iba, Y. Matsuzawa, and N. Aoyama. From Conceptual Models to Simulation Models: Model Driven Development of Agent-Based Simulations. In Proc. of the 9th Workshop on Economics and Heterogeneous Interacting Agents. Kyoto, Japan, 2004.

[9] N. R. Jennings. An agent-based approach for building complex software systems. Communications of the ACM, 44(4):35-41, 2001.

[10] MASON Home Page. George Mason University, Fairfax, VA, available at http://cs.gmu.edu/~eclab/projects/mason/.

[11] N. Minar, R. Burkhart, C. LAngton, and M. Askenazi. The Swarm Simulation System: A Toolkit for Building Multi-Agent Simulations. Working Paper 96-06-042. Santa Fe Institute, 1996.

[12] A. Molesini, A. Omicini, A. Ricci, E. Denti. Zooming multi-agent systems. In Proc of the 6th International Workshop on Agent-Oriented Software Engineering (AOSE 2005), AAMAS 2005, Utrecht, The Netherlands, 2005.

[13] M. F. Monaco, L. Moccia, and M. Sammarra. Operations Research for the management of a transhipment container terminal. The Gioia Tauro case. Maritime Economics and Logistics, 2009, Vol. 11, n. 1, pp. 7-35.

[14] M. F. Monaco, and M. Sammarra. Scheduling and dispatching models for routing straddle carriers at a container terminal. Proceedings of the XXXVIII Annual Conference of the Italian Operations Research Society, Genova, Italy, Sept. 5-8, 2007.

[15] M. J. North, T.R. Howe, N.T. Collier, and J.R. Vos. The Repast Simphony Development Environment. In Proc. of the Agent 2005 Conference on Generative Social Processes, Models, and Mechanisms, Chicago, IL , October 2005.

[16] M. J. North, T.R. Howe, N.T. Collier, and J.R. Vos. Repast Simphony Runtime System. In Proc. of the Agent 2005 Conference on Generative Social Processes, Models, and Mechanisms, Chicago, IL, October 2005.

[17] M. J. North, C. M. Macal. Managing Business Complexity: Discovering Strategic Solutions with Agent-Based Modeling and Simulation. Oxford University Press, 2007.

[18] M.J. North, E. Tatara, N.T. Collier, and J. Ozik. Visual Agent-Based Model Development with Repast Simphony. In Proc. of the Agent 2007 Conference on Complex Interaction and Social Emergence. Northwestern University, Evanston, IL, November2007.

[19] M. Resnick. Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Mi-croworlds. MIT Press, Cambridge, Mass., 1997.

[20] ROAD (Repast Organization for Architecture and Design). Repast Home Page, Chicago, IL, available as http://repast.sourceforge.net/.

[21] D. C. Schmidt. Model-Driven Engineering. IEEE Computer, 39(2):41-47, 2006.

[22] UNCTAD (2006). United Nations Conference on Trade and Development - Review of Maritime Transportation.

[23] Unified Modeling Language (UML) Specification. Version 2.1.2. Object Management Group Inc., 2007.

[24] U. Wilensky. NetLogo. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 1999.